AACL-IJCNLP 2020

The 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing

Proceedings of the Student Research Workshop

December 4 - December 7, 2020
Suzhou, China

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 978-1-952148-93-4

Introduction

Welcome to the AACL-IJCNLP 2020 Student Research Workshop (SRW)!

Held in conjunction with The 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (AACL) and the 10th International Joint Conference on Natural Language Processing conference (IJCNLP), the AACL-IJCNLP 2020 SRW is the first SRW offered in the Asia-Pacific area, and is joining its older siblings: the ACL SRW, the NAACL SRW and the EACL SRW.

SRW is a workshop for student researchers in computational linguistics and natural language processing. The SRW aims to provide student researchers the opportunity to present their work and receive constructive feedback and mentorship by experienced members of the ACL community.

Following the tradition established by previous SRWs, this year's submissions were organized into two tracks: research papers and thesis proposals.

- Research papers: Papers in this category can describe completed work, or work in progress with preliminary results. For these papers, the first author must be a current graduate or undergraduate student.
- Thesis proposals: This category is appropriate for advanced students who have decided on a thesis topic and wish to get feedback on their proposal and broader ideas for their continuing work.

We received a total of 47 submissions: 38 research papers and 9 thesis proposals. We accepted 22 research papers and two thesis proposals, resulting in an overall acceptance rate of 51%. We were delighted that submissions were diverse not only in topics but also in terms of the student demographics.

Following previous SRWs, we provided two mentoring programs for participants. The first program was the pre-submission mentoring program which offered students the opportunity to get feedback by a mentor prior to submitting their work for review. Eleven papers participated in the pre-submission mentoring program. In addition, we offered a post-accept mentoring program for accepted papers. In the post-accept mentoring program, a mentor was assigned to each accepted paper to help the student authors with the preparation of camera-ready submission and the presentation materials.

In addition to keeping up with SRW traditions, we also introduced new program features to the SRW: the **SRW Keynote** and an **SRW Best Paper Award**. We believe that these features add value to the workshop and we hope that they will be carried over to all future SRWs.

We would like to thank the 11 pre-submission mentors and the 24 post-acceptance mentors for dedicating their time and effort to help the student researchers with their paper in various stages. We would also like to thank the members of the program committee for their in-depth review and constructive feedback for each submitted paper.

Many thanks to our faculty advisors, Lun-Wei Ku and Vincent Ng, who provided tremendous help and guidance through the preparation of this workshop. Special thanks to Shruti Rijhwani for her support and great advice. We are also indebted to Emily M. Bender who accepted our invitation to give the first SRW keynote.

We would also like to thank the organizers of the AACL-IJCNLP conference for their support. Finally, we would like to thank all the student participants who have submitted their work to the workshop.

We hope you enjoy the AACL-IJCNLP 2020 SRW!

Student Chairs:

Boaz Shmueli, Academia Sinica (Taiwan)
Yin Jou Huang, Kyoto University (Japan)

Faculty Advisors:

Lun-Wei Ku, Academia Sinica (Taiwan)
Vincent Ng, University of Texas at Dallas (USA)

Pre-submission Mentors:

Valerio Basile, University of Turin
Rishi Bommasani, Stanford University
David Chiang, University of Notre Dame
Parisa Kordjamshidi, Michigan State University
Ellie Pavlick, Brown University
Sai Krishna Rallabandi, Carnegie Mellon University
Paul Rayson, Lancaster University
Arijit Sehanobish, Yale University
Anders Søgaard, University of Copenhagen
Chen-Tse Tsai, Bloomberg
Alakananda Vempala, Bloomberg
Wei Yu, Carnegie Mellon University

Post-accept Mentors:

Reinald Kim Amplayo, University of Edinburgh
Daniel Beck, University of Melbourne
Eduardo Blanco, University of North Texas
Ronald Cardenas, Charles University
Lucia Donatelli, Saarland University
Micha Elsner, The Ohio State University
Kilian Evang, DFKI
Ji He, Citadel LLC
Junjie Hu, Carnegie Mellon University
Divyansh Kaushik, Carnegie Mellon University
Shuhei Kurita, RIKEN AIP
Yang Liu, Tsinghua University
Kenton Murray, Johns Hopkins University
Denis Newman-Griffis, University of Pittsburgh
Mah Parsa, University of Toronto
Sai Krishna Rallabandi, Carnegie Mellon University
Sonit Sehanobish, Yale
Singh, Macquarie University
Haiyue Song, Kyoto University
Arseny Tolmachev, Fujitsu Labratories Ltd.
Bonnie Webber, University of Edinburgh

Pei Zhou, University of Southern California
Hao Zhu, Carnegie Mellon University
Arkaitz Zubiaga, Queen Mary University of London

Program Committee:

Piush Aggarwal, University of Duisburg-Essen, Language Technology Lab
Afroz Ahamad, BITS Pilani Hyderabad Campus
Miguel A. Alonso, Universidade da Coruña
Reem Alqifari, King Saud University & University of York
Rami Aly, University of Cambridge
Evelin Amorim, UFMG
Maria Antoniak, Cornell University
Anusha Balakrishnan, Microsoft Semantic Machines
Jorge Balazs, University of Tokyo
Esma Balkir, University of Edinburgh
Anjali Bhavan, Delhi Technological University
Tatiana Bladier, Heinrich Heine University of Düsseldorf
Arlene Casey, University of Edinburgh
Lisa Andreevna Chalaguine, UCL
Jonathan P. Chang, Cornell University
Aditi Chaudhary, Carnegie Mellon University
Jifan Chen, UT Austin
Xinchi Chen, Amazon AWS
Xiang Dai, University of Sydney
Siddharth Dalmia, Carnegie Mellon University
Samvit Dammalapati, Indian Institute of Technology Delhi
Alok Debnath, International Institute of Information Technology
Pieter Delobelle, KU Leuven
Louise Deléger, INRAE - Université Paris-Saclay
Chris Develder, Ghent University
Anne Dirkson, Leiden University
Zi-Yi Dou, Carnegie Mellon University
Hicham El Boukkouri, LIMSI, CNRS, Université Paris-Saclay
Carlos Escolano, Universitat Politècnica de Catalunya
Tina Fang, University of Waterloo
Murhaf Fares, University of Oslo
Amir Feder, Technion - Israel Institute of Technology
Dayne Freitag, SRI International
Yoshinari Fujinuma, University of Colorado Boulder
Diana Galvan-Sosa, Tohoku University
Marcos Garcia, Universidade de Santiago de Compostela
Tirthankar Ghosal, Indian Institute of Technology Patna
Arijit Ghosh Chowdhury, Manipal Institute of Technology
Abhinav Gupta, Mila
Sarah Gupta, University of Washington
Vivek Gupta, University of Utah
Barry Haddow, University of Edinburgh
Hardy Hardy, The University of Sheffield
Barbora Hladka, Charles University
Christopher Homan, Rochester Institute of Technology

Phu Mon Htut, New York University
Jeff Jacobs, Columbia University
Labiba Jahan, Florida International University
Jyoti Jha, IIIT Hyderabad
Vasu Jindal, University of Texas at Dallas
Jad Kabbara, McGill University
Tomoyuki Kajiwara, Osaka University
Zara Kancheva, IICT-BAS
Sudipta Kar, University of Houston
Alina Karakanta, University of Trento
Taiwo Kolajo, Federal University Lokoja
Mamoru Komachi, Tokyo Metropolitan University
Mandy Korpusik, Loyola Marymount University
Kalpesh Krishna, University of Massachusetts Amherst
Kemal Kurniawan, University of Melbourne
Yash Kumar Lal, Stony Brook University
Alexandra Lavrentovich, Amazon Alexa
Bowen Li, University of Edinburgh
Lei Li, Peking University
Yiyuan Li, Carnegie Mellon University
Jasy Suet Yan Liew, Syracuse University
Fangyu Liu, University of Cambridge
Di Lu, Dataminr
Sean MacAvaney, Georgetown University
Valentin Malykh, Huawei Noah's Ark Lab / Kazan Federal University
Emma Manning, Georgetown University
Zita Marinho, Priberam Labs / Institute of Systems and Robotics, IST
Bruno Martins, IST and INESC-ID
Pedro Henrique Martins, Instituto de Telecomunicações / Instituto Superior Técnico
Kate McCurdy, University of Edinburgh
Shikib Mehri, Carnegie Mellon University
Rui Meng, University of Pittsburgh
Antonio Valerio Miceli Barone, The University of Edinburgh
Tsvetomila Mihaylova, Instituto de Telecomunicações
Farjana Sultana Mim, Tohoku University
Sewon Min, University of Washington
Gosse Minnema, University of Groningen
Rohan Mishra, Delhi Technological University
Nora Muheim, University of Zürich
Masaaki Nagata, NTT Corporation
Nihal V. Nayak, Brown University
Vincent Nguyen, Australian National University & CSIRO Data61
Arturo Oncevay, The University of Edinburgh
Yasumasa Onoe, The University of Texas at Austin
Naoki Otani, Carnegie Mellon University
Endang Wahyu Pamungkas, University of Turin
Sheena Panthaplackel, The University of Texas at Austin
Chan Young Park, Carnegie Mellon University
Archita Pathak, University at Buffalo
Siyao Peng, Georgetown University
Pranav A, Dayta AI

Yusu Qian, New York University
Ivaylo Radev, IICT-BAS
Sree Harsha Ramesh, UMass Amherst
Surangika Ranathunga, University of Moratuwa
Vikas Raunak, CMU
Abhilasha Ravichander, Carnegie Mellon University
Steffen Remus, Hamburg University
Shruti Rijhwani, Carnegie Mellon University
Maria Ryskina, Carnegie Mellon University
Sepideh Sadeghi, Uber Technologies Inc.
Niloofar Safi Samghabadi, University of Houston
Jin Sakuma, University of Tokyo
Jainisha Sankhavara, DA-IICT
Ramit Sawhney, Netaji Subhas Institute of Technology
Sebastian Schuster, Stanford University
Olga Seminck, CNRS
Gautam Kishore Shahi, University of Duisburg-Essen
A.B. Siddique, University of California, Riverside
Pradyumna Sinha, Delhi Technological University
Sunayana Sitaram, Microsoft Research India
Kevin Small, Amazon
Sergey Smetanin, National Research University Higher School of Economics
Richard Sproat, Google, Japan
Shabnam Tafreshi, The George Washington University
Antonio Toral, University of Groningen
Elena Tutubalina, Kazan Federal University
Sowmya Vajjala, National Research Council
Emiel Van Miltenburg, Tilburg University
Adina Williams, Facebook, Inc.
Yumo Xu, University of Edinburgh
Rongtian Ye, Aalto University
Da Yin, Peking University
Michael Yoder, Carnegie Mellon University
Omnia Zayed, National University of Ireland Galway
Meishan Zhang, Tianjin University
Ivan Šimko, University of Zurich

Keynote Talk

Societal impacts of NLP: How and when to integrate them into your research (and how to make time for that)

Emily M. Bender
Professor, University of Washington

Abstract

In March 2020, the ACL adopted the ACM's Code of Ethics and, beginning with EMNLP 2020, ACL conferences have included ethical considerations in the review process. In this talk, I will put these developments into historical perspective, talk about the kinds of risks that give rise to them, and explore positive steps that all NLP researchers can take to increase the chances that our time, effort and creativity are put towards work that is helpful for people, especially those who have been subject to marginalization.

Bio

Dr. Emily M. Bender is a professor of linguistics at the University of Washington where she is the faculty director of the professional MS program in computational linguistics. Her research interests include the interaction of linguistics and NLP and the societal impact of language technology and how transparent documentation can help mitigate the effects of bias and the potential for trained systems to perpetuate systems of oppression. She is also actively working on how to best incorporate training on ethics and societal impact into NLP curricula.

Table of Contents

Conference Program

Dec 7, 2020 (GMT+8)

08:45–09:00 *Welcome and Opening Remarks*
Workshop chairs

09:00–10:00 Keynote Address

Societal impacts of NLP: How and when to integrate them into your research (and how to make time for that)
Emily M. Bender, University of Washington

10:00–11:00 Session SRW1: Advanced Models
Chair: Katharina Kann

Text Classification through Glyph-aware Disentangled Character Embedding and Semantic Sub-character Augmentation
Takumi Aoki, Shunsuke Kitada and Hitoshi Iyatomi

Two-Headed Monster and Crossed Co-Attention Networks
Yaoyiran Li and Jing Jiang

Towards a Task-Agnostic Model of Difficulty Estimation for Supervised Learning Tasks
Antonio Laverghetta Jr., Jamshidbek Mirzakhalov and John Licato

A Siamese CNN Architecture for Learning Chinese Sentence Similarity
Haoxiang Shi, Cen Wang and Tetsuya Sakai

Dec 7, 2020 (GMT+8) (continued)

11:00–12:00 **Session SRW2: Social Media and Applications**
Chair: Wei Emma Zhang

Automatic Classification of Students on Twitter Using Simple Profile Information
Lili-Michal Wilson and Christopher Wun

Towards Code-switched Classification Exploiting Constituent Language Resources
Kartikey Pant and Tanvi Dadu

Hindi History Note Generation with Unsupervised Extractive Summarization
Aayush Shah, Dhineshkumar Ramasubbu, Dhruv Mathew and Meet Chetan Gadoya

Unbiasing Review Ratings with Tendency Based Collaborative Filtering
Pranshi Yadav, Priya Yadav, Pegah Nokhiz and Vivek Gupta

12:00–14:00 **Break**

14:00–15:00 **Session SRW3: Low-Resource Languages**
Chair: Pranav A

Building a Part-of-Speech Tagged Corpus for Drenjongke (Bhutia)
Mana Ashida, Seunghun Lee and Kunzang Namgyal

Towards a Standardized Dataset on Indonesian Named Entity Recognition
Siti Oryza Khairunnisa, Aizhan Imankulova and Mamoru Komachi

Resource Creation and Evaluation of Aspect Based Sentiment Analysis in Urdu
Sadaf Rani and Muhammad Waqas Anwar

Dec 7, 2020 (GMT+8) (continued)

15:00–16:00 **Session SRW4: Translation and Transformation**
Chair: Yuki Arase

Making a Point: Pointer-Generator Transformers for Disjoint Vocabularies
Nikhil Prabhu and Katharina Kann

Training with Adversaries to Improve Faithfulness of Attention in Neural Machine Translation
Pooya Moradi, Nishant Kambhatla and Anoop Sarkar

Document-Level Neural Machine Translation Using BERT as Context Encoder
Zhiyu Guo and Minh Le Nguyen

A Review of Cross-Domain Text-to-SQL Models
Yujian Gan, Matthew Purver and John R. Woodward

16:00–17:00 **Session SRW5: Semantics and Pragmatics**
Chair: Vivek Gupta

Multi-task Learning for Automated Essay Scoring with Sentiment Analysis
Panitan Muangkammuen and Fumiyo Fukumoto

Aspect Extraction Using Coreference Resolution and Unsupervised Filtering
Deon Mai and Wei Emma Zhang

GRUBERT: A GRU-Based Method to Fuse BERT Hidden Layers for Twitter Sentiment Analysis
Leo Horne, Matthias Matti, Pouya Pourjafar and Zuowen Wang

Exploring Statistical and Neural Models for Noun Ellipsis Detection and Resolution in English
Payal Khullar

Dec 7, 2020 (GMT+8) (continued)

17:00–18:00 **Session SRW6: Applications and Methods**
Chair: John Licato

MRC Examples Answerable by BERT without a Question Are Less Effective in MRC Model Training
Hongyu Li, Tengyang Chen, Shuting Bai, Takehito Utsuro and Yasuhide Kawada

Text Simplification with Reinforcement Learning Using Supervised Rewards on Grammaticality, Meaning Preservation, and Simplicity
Akifumi Nakamachi, Tomoyuki Kajiwara and Yuki Arase

Label Representations in Modeling Classification as Text Generation
Xinyi Chen, Jingxian Xu and Alex Wang

Generating Inflectional Errors for Grammatical Error Correction in Hindi
Ankur Sonawane, Sujeet Kumar Vishwakarma, Bhavana Srivastava and Anil Kumar Singh

18:00–18:30 *Best Paper Award and Closing Remarks*
Workshop chairs

Text Classification through Glyph-aware Disentangled Character Embedding and Semantic Sub-character Augmentation

Takumi Aoki Shunsuke Kitada Hitoshi Iyatomi
Department of Applied Informatics, Graduate School of Science and Engineering
Hosei University, Tokyo, Japan
{takumi.aoki.4g, shunsuke.kitada.8y}@stu.hosei.ac.jp
iyatomi@hosei.ac.jp

Abstract

We propose a new character-based text classification framework for non-alphabetic languages, such as Chinese and Japanese. Our framework consists of a variational character encoder (VCE) and character-level text classifier. The VCE is composed of a β-variational auto-encoder (β-VAE) that learns the proposed glyph-aware disentangled character embedding (GDCE). Since our GDCE provides zero-mean unit-variance character embeddings that are dimensionally independent, it is applicable for our interpretable data augmentation, namely, semantic sub-character augmentation (SSA). In this paper, we evaluated our framework using Japanese text classification tasks at the document- and sentence-level. We confirmed that our GDCE and SSA not only provided embedding interpretability but also improved the classification performance. Our proposal achieved a competitive result to the state-of-the-art model while also providing model interpretability.

1 Introduction

Some Asian languages (e.g., Chinese and Japanese) use *glyphs* to give visual meaning to characters. For example, the following Japanese characters have a common form of "辶," which is a sub-character meaning of the related word *road*: "迫" (approach: come near the destination by *road*) and "追" (follow: track the *road*). In consideration of these characteristics of the language, several glyph-aware natural language processing (NLP) models have been proposed (Shimada et al., 2016; Liu et al., 2017; Kitada et al., 2018; Sun et al., 2019). These deep-learning-based models train input text as a sequence of character images and learn character embeddings from the images.

In general, the interpretability of the NLP model is important in terms of its reliability, as well as providing the required performance for the task. If imaged-based models can learn these sub-characters in a way that is interpretable, it helps greatly in improving the overall interpretability of the models.

In terms of improving the interpretability of models, disentangled representation learning method has received a great deal of attention in recent years, such as InfoGAN (Chen et al., 2016) and β-variational auto-encoder (β-VAE) (Higgins et al., 2017). This learning method transforms the input data into low-dimensional representations that are independent of each other while still retaining the important content. Although it has been actively discussed in the field of computer vision, there are few applications in the field of NLP.

In terms of ensuring model robustness, data augmentation is necessary and essential in machine learning today. With regard to this desirable feature, glyph-aware embedding (i.e., image-based character embedding) allows data augmentation without contextual consideration, such as word dropout (Iyyer et al., 2015) and wildcard training (Shimada et al., 2016). Simple data augmentation based on dropout does not consider the features of the input space. If the NLP method based on glyph-aware embedding is highly interpretive, such as a disentangled representation, an effective data augmentation method can be achieved. This improves not only the robustness of the model but also its interpretability.

In this paper, we propose a general-purpose text classification framework that gives interpretability to data augmentation for image-based glyph-aware character embedding, which has the various advantages mentioned above. The framework consists of two novel methods: (1) glyph-aware disentangled character embedding (GDCE) and (2) semantic sub-character augmentation (SSA). Each method has the following simple but effective features:

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 1–7
December 4 - 7, 2020.

- The GDCE is obtained from the variational character encoder (VCE), which is the encoder part of the β-VAE. The VCE takes advantage of the β-VAE to create a low-dimensional representation of the characters, where each dimension follows an independent normal distribution. Therefore, the GDCE provides a disentangled character embedding in which each of the dimensions corresponds to the structure of the sub-character.

- The SSA alters only one dimension of the GDCE, which corresponds with altering some part of the shape of the original character, and can present how the character has changed. In other words, these combinations are equivalent to replacing the sub-character of a character with another readable sub-character.

Our framework improves the interpretability of character embedding by the GDCE, and the SSA provides interpretable data augmentation suitable for the GDCE. We verified the text classification ability of our proposed framework using Japanese text classification tasks. [1]

2 Related work

2.1 Glyph-aware Natural Language Processing

Embedding methods based on character images have been proposed with some excellent success (Chen et al., 2015; Sun et al., 2016; Yu et al., 2017; Sun et al., 2019; Dai and Cai, 2017; Shimada et al., 2016; Liu et al., 2017; Kitada et al., 2018; Ke and Hagiwara, 2017; Aldón Mínguez et al., 2016). These methods are also called glyph-aware embedding as they generate embeddings that take into account the shape of the characters or sub-characters. These image-based methods mainly use convolutional neural networks (CNNs) or convolutional auto-encoders (CAEs) (Masci et al., 2011) for character-embedding learning, and they perform well because of the following advantages: (1) they operate without the cumbersome word segmentation required by some Asian languages, and (2) they can apply additional image-based data augmentation.

[1] The code required to reproduce the experiments is available on GitHub. https://github.com/IyatomiLab/GDCE-SSA

2.2 Data Augmentation for Natural Language Processing

For NLP tasks, it is challenging to apply data augmentation methods because of the need to consider the context of the text (Sennrich et al., 2016; Jia and Liang, 2016; Silfverberg et al., 2017; Edunov et al., 2018). Several data augmentation methods that do not require text analysis have been proposed for word embedding (Iyyer et al., 2015; Zhang et al., 2016) and character embedding (Shimada et al., 2016). In particular, Shimada et al. (2016) achieved significant performance improvements by applying dropout (Hinton et al., 2012)-based data augmentation to a type of character embedding called wildcard training (WT). However, these methods have little interpretability of what the data augmentation means in the input text, partly due to the lack of interpretability of the embedding itself. Our proposed SSA is improved WT, and it replaces the sub-character of a character with another readable sub-character.

2.3 Learning Interpretable Character Embeddings

For learning a latent representation that can be interpreted, InfoGAN (Chen et al., 2016) and β-VAE (Higgins et al., 2017) are well known. Unlike InfoGAN, β-VAE is stable while training, requires less assumptions about the data, and relies on only a single hyperparameter β. Because of these advantages, several improved models based on β-VAE have been proposed (e.g., Factor-VAE (Kim and Mnih, 2018), HFVAE (Esmaeili et al., 2019)). Therefore, in this paper, we use β-VAE as a VCE to learn interpretable character embeddings.

3 Methodology

In this paper, we propose a new character-based text classification framework that includes a new character embedding method, consisting of glyph-aware disentangled character embedding (GDCE) and semantic sub-character augmentation (SSA). Figure 1 shows an overview of the proposed text classification framework.

3.1 Glyph-aware Disentangled Character Embedding (GDCE)

We obtain the GDCE using the VCE based on the β-VAE. Since the GDCE provides dimensionally independent features, we expect to solve the prob-

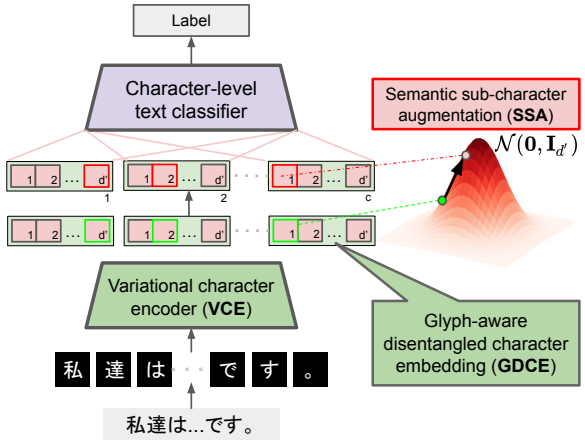


Figure 1: Overview of our text classification framework. Each character in the target text is transformed to an image and forwarded as a glyph feature to the subsequent VCE. The VCE is composed of a β-VAE, and it learns the proposed GDCE. Owing to the attractive properties of the GDCE, character-level text classifier can take advantage of the interpretable and highly effective data augmentation method, SSA.

lem of the poorly interpretable character embedding obtained by the CAE.

β-VAE is a generative model that estimates the data distribution $p(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ is a d-dimensional input. Let $\boldsymbol{z} \in \mathbb{R}^{d'}$ be a d'-dimensional latent variable, which is derived from the GDCE in this paper; $p(\boldsymbol{z})$ is a normal distribution, which is the prior distribution of the latent variables, $q(\boldsymbol{z}|\boldsymbol{x})$ is the posterior distribution, and $p(\boldsymbol{x}|\boldsymbol{z})$ is a generative model. We optimize the following function:

$$\begin{aligned}\mathcal{L}_{\beta\text{-VAE}} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \\ - \beta D_{\mathrm{KL}}[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})],\end{aligned} \tag{1}$$

where β is a balancing coefficient for the second term. The first term represents the reconstruction error of the character image. The second term represents the regularization of the latent variables that are learned so as to follow the prior distribution by the KL divergence $D_{\mathrm{KL}}[\cdot||\cdot]$. If the coefficient β increases, it is possible to obtain a representation of the features where each dimension is independent (Higgins et al., 2017).

However, the latent variables themselves are a probability distribution and cannot be backpropagated to the encoder. Hence, the reparameterization trick (Kingma and Welling, 2013) of the approximation method is used. We let $\boldsymbol{\alpha}$ be a sampled random variable from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d'})$ and calculate the latent variables as follows:

$$\boldsymbol{z} = \mu(\boldsymbol{x}) + \boldsymbol{\alpha} \odot \sigma(\boldsymbol{x}), \quad \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d'}), \tag{2}$$

where $\odot$ is an element-wise product, μ is the mean of the distribution, and σ is the variance of the distribution. Here, $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are d'-dimensional vectors obtained from the β-VAE.

3.2 Character-level Text Classification with Semantic Sub-character Augmentation (SSA)

The sequence of c embedded characters $C = \{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(c)}\}$ from the GDCE, where $\boldsymbol{z}^{(t)}$ is the t-th character embedding, in the VCE are provided to the following character-level text classifier. The parameters of the classifier are optimized in the back-propagation using the cross-entropy error.

In this paper, we propose SSA as a data augmentation method. Taking advantage of the preferred features of the embedding created by the GDCE, we expect that the sub-character of a character will be replaced by another readable sub-character, using the SSA.

Let γ be the perturbation range, and the formula of the SSA for the i-th dimension $\boldsymbol{z}_i^{(t)}$ of the character embedding $\boldsymbol{z}^{(t)}$ is defined as follows:

$$\boldsymbol{z}_i'^{(t)} = \boldsymbol{z}_i^{(t)} + u, \quad u \sim \mathcal{U}(-\gamma, \gamma), \tag{3}$$

where $u \sim \mathcal{U}(a, b)$ indicates that the random variable u has a uniform distribution with the minimum a and the maximum b. Since each dimension of the GDCE follows $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d'})$, the character embedding converted in Eq. 3 falls within the range of trained character-embedding values.

4 Experiment Settings

4.1 Evaluation Datasets

We evaluated our framework with the following datasets: newspaper and livedoor. These datasets were split into two parts: 80% for training and 20% for evaluation. Because these datasets contain new words and/or meanings related to current affairs, accurate word segmentation through morphological analysis has been a challenge in conventional word-level processing for Japanese. Therefore, we can avoid such difficulties by using character-level input instead of word-level input[2].

Newspaper. The newspaper dataset used in Shimada et al. (2016) contains 5,610 Japanese major web newspaper articles (Asahi, Mainichi, Sankei,

[2]It is generally known that a character-level model performs better than a word-level model in Chinese and Japanese (Zhang and LeCun, 2017).

Layer	Encoder
1	Conv2d ($k = (4, 4)$, $o = 32$, $s = 2$) $\rightarrow$ ReLU
2	Conv2d ($k = (4, 4)$, $o = 32$, $s = 2$) $\rightarrow$ ReLU
3	Conv2d ($k = (4, 4)$, $o = 64$, $s = 2$) $\rightarrow$ ReLU
4	Conv2d ($k = (4, 4)$, $o = 64$, $s = 2$) $\rightarrow$ ReLU
5	Linear($o = 256$) $\rightarrow$ ReLU
6	Linear($o = 2 \times 10$)

Layer	Decoder
1	Linear ($o = 256$) $\rightarrow$ ReLU
2	Linear ($o = 1024$) $\rightarrow$ ReLU
3	Deconv2d ($k = (4, 4)$, $o = 64$, $s = 2$) $\rightarrow$ ReLU
4	Deconv2d ($k = (4, 4)$, $o = 32$, $s = 2$) $\rightarrow$ ReLU
5	Deconv2d ($k = (4, 4)$, $o = 32$, $s = 2$) $\rightarrow$ ReLU
6	Deconv2d ($k = (4, 4)$, $o = 1$, $s = 2$) $\rightarrow$ Sigmoid

Table 1: Architecture of β-VAE. Kernel size k, output size o, and stride size s was set to the above table.

and Yomiuri) in the categories of politics, the economy, and international news, for a total of 22,440 articles.

Livedoor. The livedoor dataset is commonly used to evaluate models for Japanese.[3] The dataset contains, for example, 870 and 900 Japanese sentences in the categories of movie-enter and sports-watch, respectively. In all the nine categories, it contains a total of 7,367 articles.

4.2 Model Architectures

We trained the VCE based on β-VAE and character-level CNN (CLCNN) (Zhang et al., 2015) as text classifier independently. The hyperparameters of these models were adjusted with a validation set split from the training set, and the predicted results of the evaluation set were reported.

β-variational auto-encoder (β-VAE). Table 1 shows the architecture of β-VAE. Generally, training of β-VAE is unstable, and requires adjustment of hyperparameters. In this paper, we carefully tuned hyperparameters based on Locatello et al. (2019). Adam (Kingma and Ba, 2014) was used to maximize $\mathcal{L}_{\beta\text{-VAE}}$, as shown in Eq. 1. We set train batch size to 64 and the learning rate to 1e-4.

To obtain the GDCE, we trained the VCE with 6,631 common Japanese characters, including Japanese Hiragana, Katakana, and Kanji[4], as well as English alphabets and symbols. These characters were converted to $d = 64 \times 64$ grayscale

[3] https://www.rondhuit.com/download.html#ldcc

[4] From the Japanese Industrial Standards; first and second levels.

Layer	CLCNN
1	Conv1d ($k = 3$, $o = 512$) $\rightarrow$ ReLU
2	Maxpool1d ($k = 3$, $s = 3$)
3	Conv1d ($k = 3$, $o = 512$) $\rightarrow$ ReLU
4	Maxpool1d ($k = 3$, $s = 3$)
5	Conv1d ($k = 3$, $o = 512$) $\rightarrow$ ReLU
6	Conv1d ($k = 3$, $o = 512$) $\rightarrow$ ReLU
7	Linear ($o =$ #classes)

Table 2: Architecture of CLCNN. Kernel size k, output size o, and stride size s was set to the above table.

character images and used as input $\boldsymbol{x}$ to the VCE. We set $\beta = 8$ and $d' = 10$ for all tasks, $\gamma = 1.5$ for the newspaper, and $\gamma = 2.0$ for the livedoor.

Character-level convolutional neural network (CLCNN). Table 2 shows the architecture of CLCNN. We trained CLCNN with the same parameters as in Shimada et al. (2016). Similar to training the character embedding model, Adam was used to minimize the cross-entropy error. We set the learning rate of Adam to 1e-4 and weight decay to 1e-4, train batch size to 256 for the livedoor, and 512 for the newspaper.

In training the CLCNN, we used the GDCE results obtained by the VCE as the input. For training, $c = 128$ consecutive characters were extracted from the text in the newspaper, and $c = 80$ consecutive characters were extracted from the title text in the livedoor. For evaluation, in the newspaper, $c = 128$ characters were slid one by one, the entire text was used as input in the same manner as in Shimada et al. (2016); in the livedoor, it was the same as in the training.

5 Results and Discussion

First, as a comparison of embedding methods, we compared the GDCE with the conventional CAE-based embedding (Shimada et al., 2016). Second, as a comparison of data augmentation methods for image-based character embedding, we also compared the proposed SSA with the conventional WT, the latter of which has reported excellent results but offers no way of interpreting the change on the embedding space.

5.1 Effectiveness of the Proposal on Text Classification

Table 3 presents a comparison of the proposed GDCE and CAE-based embedding. The GDCE showed better document- and sentence-level classification performance than the conventional CAE-

	Accuracy [%]					
	Newspaper			Livedoor		
+ CLCNN	Vanilla	+ WT	+ SSA (**Ours**)	Vanilla	+ WT	+ SSA (**Ours**)
VCE (**Ours**)	**81.02**	82.78	†**84.00**	**67.16**	68.59	†**69.05**
CAE	‡79.81	‡81.62	81.35	‡58.39	‡60.87	60.53

Table 3: A comparison between the VCE (with proposed GDCE) and the CAE in the newspaper and the livedoor results. We compared our proposed framework (presented as †; a disentangled representation) with the state-of-the-art framework of Shimada et al. (2016) (presented as ‡; without the consideration of disentangled representation). Our proposed framework had the highest performance. The model using the VCE performed better than the CAE.

based character embedding without data augmentation. This may be due to the fact that the characters to be learned by the VCE are distributed in a limited embedded space centered on zeros, so the later stage of the CLCNN training became more effective. The WT, which randomly set all representations of a particular character embedding to zero, enhanced the discrimination of both models. The effect on the CAE-based model was particularly large, as reported in previous studies. We can confirm an effect of the WT as a dropout for preventing overfitting, but it did not provide an interpretation of what was changed in the character embeddings.

The proposed SSA provided us with an idea of what the embedding changes would look like, while also providing the same discriminatory capacity as the WT. This may be due to the fact that the GDCE had standardized metrics in the embedding space (i.e., the embedding had a normal distribution), so that the distances between the character embeddings were within the range of what could be assumed. Hence, the size of the perturbations applied could be designed, allowing for meaningful data augmentation. However, the CAE with SSA did not show an improved classification performance. This may be due to the fact that the CAE with SSA does not change to a meaningful character representation.

5.2 Effectiveness of the Proposal on Interpretation

Figure 2 shows a comparison of the reconstructed character images when a $\pm 2.0\sigma$ perturbation is placed on the 2a (the GDCE) and 2b character embedding obtained by the CAE. In Figure 2a, it is confirmed that the shape of the character replaced a different interpretable character or characters with a similar different subcomponent in the input space. In particular, by adding a perturbation to the fifth dimension of the embedding of "迫" or "追" (containing a sub-char. of "辶," meaning *road*), it can be interpreted that it changed to "氵" (sub-char. of *water*) or "辶" (sub-char. of *road*, the same as "辶"). In addition, by adding a perturbation to the first dimension of the embedding of "綱" or "縄" (containing a sub-char. of "糸," meaning *yarn*), it can be interpreted that it changed to "扌" (sub-char. of *hand*) or "金" (sub-char. of *gold*). From these results, we are convinced that such a replacement in the embedding resulted in more effective data augmentation for training the model.

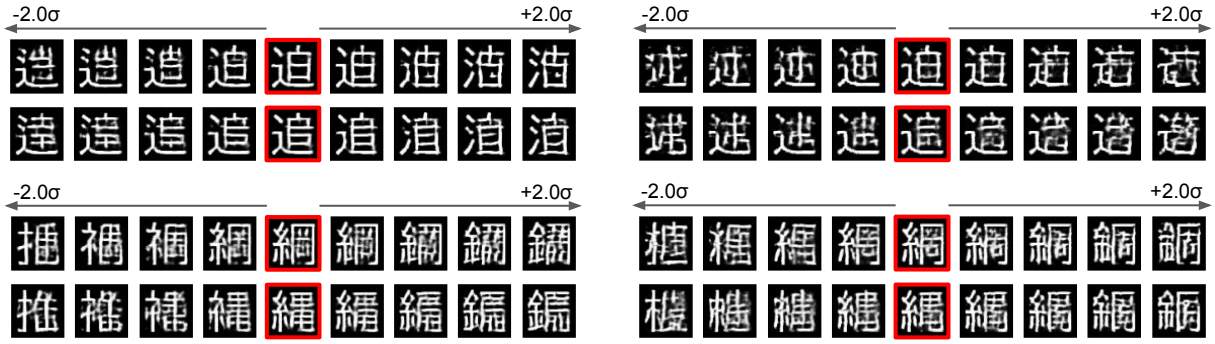
As seen in Figure 2b, in contrast, we were unable to identify these trends. We consider this is one of the typical benefits of our framework in that each dimension of the GDCE is independent and each of them affects each character component (e.g., sub-char. or radical of the character) with independence. In other words, we can change only some part of the character by changing certain dimensions of the embedding.

Since the SSA is a local transformation for the parts of the character shown above, even some characters that do not actually exist are generated by the combination of parts. These are not readable as *correct* characters, but we can make certain interpretations of them. In sum, the combination of the proposed GDCE and SSA provides us with the interpretability of the data augmentation as well as embedding the character while providing a high discriminative power.

5.3 The Effect of Hyperparameters

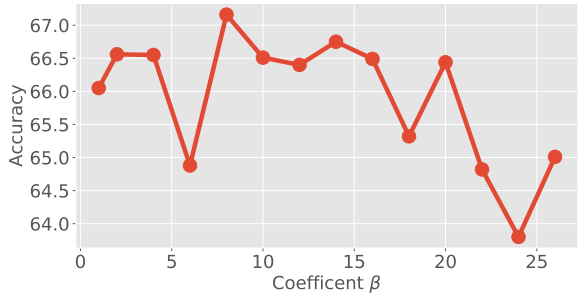
To understand the effect of hyperparameters, we analyzed the coefficient β and perturbation size γ using the livedoor, as shown in Figure 3.

The effect of coefficient β. Figure 3a shows the effect of coefficient β on the evaluation performance with $\gamma = 0$ (i.e., without SSA). In our experiments, we confirmed that $\beta = 8$ is the best from

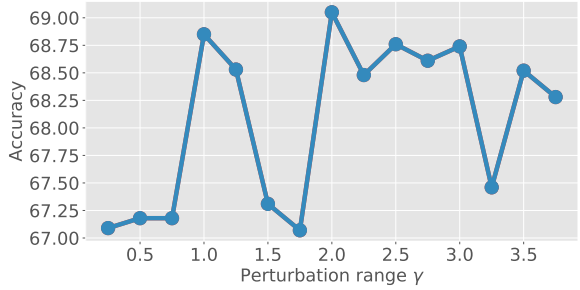


(a) VCE (**proposed**) (b) CAE

Figure 2: The results of reconstructing character images from the character embedding trained by the VCE and CAE with perturbation added between 2.0σ. **The upper side** is the reconstructed image of "迫" (approach) and "追" (follow). In the reconstruction from the embedding by the VCE and by adding noise to the fifth dimension of the embedding of "迫" or "追" (containing a sub-char. of "辶" meaning *road*), it can be interpreted that it changed to "氵" (sub-char. of *water*) or "辶" (sub-char. of *road*, the same as "辶"). **The lower side** is the reconstructed image of "綱" (rope) and "縄" (cord). In the reconstruction from the embedding by the VCE and by adding noise to the first dimension of the embedding of "綱" or "縄" (containing a sub-char. of "糸" meaning *yarn*), it can be interpreted that it changed to "扌" (sub-char. of *hand*) or "金" (sub-char. of *gold*).



(a) The effect of coefficient β ($\gamma = 0$ i.e., without SSA).



(b) The effect of perturbation range γ ($\beta = 8$).

Figure 3: The effect of hyperparameters in our framework using the livedoor dataset on the evaluation performance.

the viewpoint of disentanglement and accuracy.

The effect of perturbation size γ. Figure 3b shows the effect of perturbation range γ in SSA on the evaluation performance with $\beta = 8$. Based on the notion that each dimension of the target character embedding follows $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d'})$, the perturbation range γ was chosen to be from 1.0σ (covering 68% of the distribution) to 3.0σ (covering almost the entire distribution). The best performance was obtained when the perturbation range was set to $\gamma = 2.0$. This suggests that the character embedding trained by the VCE followed a normal distribution with a mean of $\mu = 0$ and a standard deviation of $\sigma = 1.0$. To cover the distribution, it is considered useful to add perturbation in the range of $\gamma = 2.0$ corresponding to 2.0σ (covering 95% of the distribution).

5.4 Limitations of the Current Study

At present, the role of each dimension in the character reconstruction of the GDCE cannot be clearly defined because it depends on the training of the model. Also, since the VCE was independently trained from the classifier (i.e., not in an end-to-end manner), trained embedding can only consider visual features, not the semantic ones. We will be working on these in the future.

6 Conclusion

We propose a new character-based text classification framework for non-alphabetic languages. As the name implies, the combination of our GDCE and SSA not only provided embedding interpretability but also improved the text classification performance. Our GDCE provided better text classification performance than conventional CAE-based character embedding without data augmentation. Finally, our framework achieved a competitive result to the conventional state-of-the-art CAE-based embedding with WT while also providing model interpretability.

References

David Aldón Mínguez, Marta Ruiz Costa-Jussà, and José Adrián Rodríguez Fonollosa. 2016. Neural machine translation using bitmap fonts. In *Proc. of EAMT HyTra Workshop*, pages 1–9.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. of NIPS*, pages 2172–2180.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Proc. of ICOAI*, pages 1236–1242.

Falcon Z Dai and Zheng Cai. 2017. Glyph-aware Embedding of Chinese Characters. In *Proc. of SCLeM Workshop*, pages 64–69.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding Back-Translation at Scale. In *Proc. of EMNLP*, pages 489–500.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. 2019. Structured Disentangled Representations. In *Proc. of AISTAT*, pages 2525–2534.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *In Proc. of ICLR*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR preprint arXiv:1207.0580*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proc. of ACL-IJCNLP*, pages 1681–1691.

Robin Jia and Percy Liang. 2016. Data Recombination for Neural Semantic Parsing. In *Proc. of ACL*, pages 12–22.

Yuanzhi Ke and Masafumi Hagiwara. 2017. Radical-level Ideograph Encoder for RNN-based Sentiment Analysis of Chinese and Japanese. In *Proc. of ACML*, pages 561–573.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by Factorising. In *Proc. of ICML*, pages 2649–2658.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR preprint arXiv:1312.6114*.

Shunsuke Kitada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2018. End-to-End Text Classification via Image-based Embedding using Character-level Networks. In *Proc. of IEEE AIPR Workshop*, pages 1–4.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning Character-level Compositionality with Visual Features. *In Proc. of ACL*, pages 2059–2068.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *Proc. of ICML*, pages 4114–4124.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *Proc. of ICANN*, pages 52–59.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proc. of ACL*, pages 86–96.

Daiki Shimada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2016. Document classification through image-based character embedding and wildcard training. In *Proc. of IEEE Big Data Workshop*, pages 3922–3927.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proc. of CoNLL-SIGMORPHON*, pages 90–99.

Chi Sun, Xipeng Qiu, and Xuan-Jing Huang. 2019. VCWE: Visual Character-Enhanced Word Embeddings. In *Proc. NAACL-HLT*, pages 2710–2719.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Inside out: Two jointly predictive models for word representations and phrase representations. In *Proc. of AAAI*.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint Embeddings of Chinese Words, Characters, and Fine-grained Subcharacter Components. In *Proc. of EMNLP*, pages 286–291.

Dongxu Zhang, Tianyi Luo, and Dong Wang. 2016. Learning from LDA Using Deep Neural Networks. In *Proc. of NLPCC-ICCPOL*, pages 657–664.

Xiang Zhang and Yann LeCun. 2017. Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean? *CoRR preprint arXiv:1708.02657*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*, pages 649–657.

Two-Headed Monster And Crossed Co-Attention Networks

Yaoyiran Li
Language Technology Lab, TAL
University of Cambridge
yl711@cam.ac.uk

Jing Jiang
Living Analytics Research Centre
Singapore Management University
jingjiang@smu.edu.sg

Abstract

This paper investigates a new co-attention mechanism in neural transduction models for machine translation tasks. We propose a paradigm, termed Two-Headed Monster (THM), which consists of two symmetric encoder modules and one decoder module connected with co-attention. As a specific and concrete implementation of THM, Crossed Co-Attention Networks (CCNs) are designed based on the Transformer model. We test CCNs on WMT 2014 EN-DE and WMT 2016 EN-FI translation tasks and show both advantages and disadvantages of the proposed method. Our model outperforms the strong Transformer baseline by 0.51 (big) and 0.74 (base) BLEU points on EN-DE and by 0.17 (big) and 0.47 (base) BLEU points on EN-FI but the epoch time increases by circa 75%.

1 Introduction

Attention has emerged as a prominent mechanism extensively adopted in neural modules in a wide range of research problems (Das et al., 2017; Hermann et al., 2015; Rocktäschel et al., 2015; Santos et al., 2016; Xu and Saenko, 2016; Yang et al., 2016; Yin et al., 2016; Zhu et al., 2016; Xu et al., 2015; Chorowski et al., 2015) such as VQA, reading comprehension, textual entailment, image captioning and speech recognition. Its remarkable success is also embodied in machine translation tasks (Bahdanau et al., 2014; Vaswani et al., 2017).

This work proposes an end-to-end co-attentional neural structure named Crossed Co-Attention Networks (CCNs) to address machine translation, a typical sequence-to-sequence NLP task. We customize the transformer (Vaswani et al., 2017) featured by non-local operations (Wang et al., 2018) with two input branches and tailor the transformer's multi-head attention mechanism to the needs of information exchange between these two parallel branches.

A higher-level and more abstract paradigm generalized from CCNs is denoted as "Two-Headed Monster" (THM), representing a broader class of neural structures benefiting from two parallel neural channels that would be intertwined with each other through co-attention mechanism as illustrated in Fig. 1.

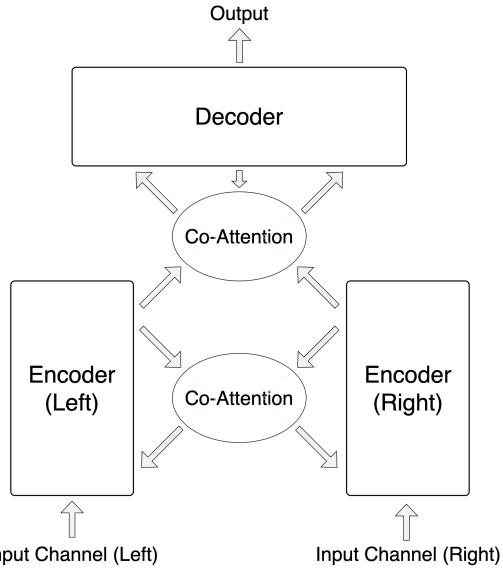


Figure 1: Two-Headed Monster.

Needless to say, co-attention is widely adopted in multi-modal scenarios (Lu et al., 2016a; Yu et al., 2017; Tay et al., 2018; Xiong et al., 2016; Lu et al., 2016b), the basic idea of which is to make two feature maps from different domains to attend to each other and thus output summarized representations for each domain. In this work, we emphasize a parallel and symmetric manifold operating on two input channels and possessing two output channels but do not assume that the two channels of input must be disparate. Our co-attention mechanism is designed in a "Transformer" style, and to the best of our knowledge, our proposed Crossed Co-Attention Network is one of

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 8–15

the first implementations of co-attention on transformer model (Lu et al., 2019; Tan and Bansal, 2019) and we are the first to apply transformer-based co-attention on machine translation tasks. In particular, we apply our model on the popular WMT machine translation tasks where two input channels are in one same domain. Our code also leverages half-precision floating-point format (FP16) training and synchronous distributed training for inter-GPU communication (we do not discard gradients calculated by "stragglers") which dramatically accelerate our training procedure (Ott et al., 2018; Micikevicius et al., 2018).

2 Model Architecture

In this section, we first define co-attention as a generic concept, following Wang et al. (2018)'s definition of non-local operation. After that, we propose an end-to-end neural architecture based on the transformer to address machine translation tasks where the model takes input from two channels. In particular, we design a Crossed Co-Attention Mechanism to make our model capable of attending to two parallel information flows simultaneously in both the encoding and the decoding stages. Our co-attention mechanism is naively realized by a crossed connection of Value (V), Key (K) and Query (Q) gates of a regular multi-head attention module, so we term our model Crossed Co-Attention Networks.

2.1 Generic Co-Attention

In this section, we first review a non-local operation and bridge it to the dot-product attention that is widely used in self-attention modules and then formulate the co-attention mechanism in a generic way. A non-local operation is defined as a building block in deep neural networks that captures long-range dependencies where every response is computed as a linear combination of all features in the input feature map (Wang et al., 2018). Suppose the input feature maps are $V = [v_1, v_2, \dots, v_n]^T \in \mathbb{R}^{n\times d}$, $K = [k_1, k_2, \dots, k_n]^T \in \mathbb{R}^{n\times d}$ and $Q = [q_1, q_2, \dots, q_n]^T \in \mathbb{R}^{n\times d}$ and the output feature map $Y = [y_1, y_2, ..., y_n]^T \in \mathbb{R}^{n\times d}$ is of the same size as the input. Then a generic non-local operation is formulated as follows:

$$y_i = \frac{1}{C(q_i, K)} \sum_j f(q_i, k_j) g(v_j). \quad (1)$$

We basically follow the definition of no-local operation by Wang et al. (2018) where $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a pairwise function ("$\times$" is Cartesian product), $g : \mathbb{R}^d \to \mathbb{R}^d$ is a unary function and $C : \mathbb{R}^d \times \mathbb{R}^{n\times d} \to \mathbb{R}$ calculates a normalizer, but dispense with the assumption that $V = K = Q$. However, if we assume $f(q_i, k_j) = e^{(q_i^T W^Q)\cdot(k_j^T W^K)^T}$, $g(v_i) = v_i^T W^V$, the normalizer $C(q_i, K) = \sum_j f(q_i, k_j)$ and $V = K = Q$, then the non-local operation degrades to the multi-head self-attention as is described in (Vaswani et al., 2017) (formula 2 describes only one attention head):

$$Y = \text{softmax}(QW^Q(KW^K)^T)VW^V. \quad (2)$$

Considering two input channels, denoted as 'left' and 'right', we present the following operation as a definition of a generic co-attention where $\alpha(\cdot), \beta(\cdot)$ define if the input to V,K and Q input gates are from 'left' or 'right' encoder branches, or 'decoder' branch respectively, directing the connections of co-attention.

$$y_i^{\text{left}} = \frac{1}{C^{\text{left}}(q_i^{\alpha(Q)}, K^{\alpha(K)})} \cdot \sum_j f^{\text{left}}(q_i^{\alpha(Q)}, k_j^{\alpha(K)}) g^{\text{left}}(v_j^{\alpha(V)}), \quad (3)$$

$$y_i^{\text{right}} = \frac{1}{C^{\text{right}}(q_i^{\beta(Q)}, K^{\beta(K)})} \cdot \sum_j f^{\text{right}}(q_i^{\beta(Q)}, k_j^{\beta(K)}) g^{\text{right}}(v_j^{\beta(V)}). \quad (4)$$

Under the umbrella of Two-Headed Monster (THM), as in Fig. 1, for the co-attention serving encoders, $\alpha(\cdot), \beta(\cdot)$ take value from {'left', 'right'} and for encoder-decoder co-attention, they take value from {'left', 'right', 'decoder'}. Note that when $\alpha(\cdot)$ = 'left', $\beta(\cdot)$ = 'right' the co-attention degrades to two self-attention modules in Transformer encoders. Another example of $\alpha(\cdot), \beta(\cdot)$ is a crossed connection which we will introduce in Section 2.2 as illustrated in Fig. 2. In its encoder's co-attention, $\alpha(V)$ = $\alpha(K)$ = 'left' and $\alpha(Q)$ = 'right', $\beta(V)$ = $\beta(K)$ = 'right' and $\beta(Q)$ = 'left'. For the encoder-decoder co-attention, however, $\alpha(V)$ = 'left', $\alpha(K)$ = 'right', $\beta(V)$ = 'right', $\beta(K)$ = 'left' and $\alpha(Q)$ = $\beta(Q)$ = 'decoder'.

2.2 Crossed Co-Attention Networks

Our implementation of CCN, based on Transformer, consists of two symmetrical branches, working in parallel. Different from previously

Figure 2: Crossed Co-Attention Networks.

known co-attentions such as (Xiong et al., 2017; Lu et al., 2016a), our design is built through connecting two multiplicative attention modules (Vaswani et al., 2017), each containing three gates, i.e., V, K and Q. The information flows from two input channels and then interact with and benefit from each other via crossed connections. Suppose the input fed into the left branch is X_{left}, and the right branch X_{right}. In our encoder, the left branch takes input from X_{left} as V and K and takes the input X_{right} as Q. The right branch, however, takes the input X_{left} as Q and X_{right} as V and K (each way of connection corresponds to a choice of $\alpha(\cdot), \beta(\cdot)$). This design is, in a sense, meant for the two branches to relatively keep the information in their own domains. A special case is, if $g(v_i) = v_i$, then the response y_i will be in the row space of V. Because when an attention takes input V from its own branch, the output responses will, by and large, carry the information of the branch. In machine translation tasks, the two encoder branches take in one same input sequence, but in order to reduce the redundancy of two parallel branches, we apply dropout and input corruption on input embeddings for two branches respectively. While our model shares BPE embeddings (Sennrich et al., 2015) globally, we randomly swap two sub-word tokens in the input matrices at a probability of 0.5.

In the encoder-decoder co-attention layers, the multi-head attention on two decoder branches uses the output from two encoder branches as V and K alternatively while taking in the output of the self-attention layer in the decoder as Q. The output of the two branches in the decoder is processed through concatenation, linear transformation and then fed into a feed-forward network. The self-attention layer in the decoder is also used for reading shifted output embedding into the network. We adopt the same input masking and sinusoidal position encoding as the Transformer, which will not be expanded here.

Although we connect the attention gates the way we describe above, there are plenty of other choices of connections for the co-attention when $\alpha(\cdot), \beta(\cdot)$ vary. We find that some different selections may produce similarly good results on some NMT datasets but this work, as a preliminary investigation, only demonstrates the case specified above in Fig. 2 but leaves the study of how different ways of connections cater to different tasks for our future work.

Model	Dataset	Epoch Time (s)	BLEU	Number of Parameters	Batch Size
Transformer-Base	WMT2014 EN-DE	**684.52**	27.21	61,364,224	6,528
THM / CCN-Base	WMT2014 EN-DE	1090.65	**27.95**	114,928,640	6,528
Transformer-Base	WMT2016 EN-FI	**232.97**	16.12	55,883,776	6,528
THM / CCN-Base	WMT2016 EN-FI	410.79	**16.59**	109,448,192	6,528
Transformer-Big	WMT2014 EN-DE	**1982.63**	28.13	210,808,832	2,176
THM / CCN-Big	WMT2014 EN-DE	3611.53	**28.64**	424,892,416	2,176
Transformer-Big	WMT2016 EN-FI	**726.51**	16.21	199,847,936	2,176
THM / CCN-Big	WMT2016 EN-FI	1387.22	**16.38**	413,931,520	2,176

Table 1: Comparisons between our proposed method and Transformer baseline on WMT 2014 EN-DE and WMT 2016 EN-FI.

3 Experiments

3.1 Setup

We demonstrate our model on WMT 2014 EN-DE and WMT 2016 EN-FI machine translation tasks. For convenience, in this section, we do not differentiate between the notion of THM and CCN (which is an implementation of THM). The raw input data is pre-processed with length filtering as previous work (Ott et al., 2018). Our final dataset consists of $4,575,637$ training examples, $3,000$ validation examples and $3,003$ test examples for EN-DE, and $2,073,194$ training examples, $1,500$ validation examples and $3,000$ test examples for EN-FI. Considering the scale of the training sets, we adopt shared BPE dictionaries of size $33,712$ for EN-DE and $23,008$ for EN-FI. Our CCNs are established with 6 encoder and decoder blocks and a hidden state of size 512 for base models and with also 6 such blocks but a hidden state of $1,024$ neurons for big models. That exactly corresponds to the settings of the Transformer paper (Vaswani et al., 2017). We train our models on an NVIDIA DGX-1 GPU server with 4 TESLA V100-16GB GPUs. In order to make full use of the computational resources, FP16 computation is adopted, and we use a batch size of $6,528$ tokens/GPU for base models and $2,176$ for big models (both Transformer and THM). We adopt the Sequence-to-Sequence Toolkit FairSeq (Ott et al., 2019) released by the Facebook AI Research for our Transformer baseline[1], upon which our THM code is built as well. We train all base models for around one day and big models for around two days. For model selection, we strictly choose the model that achieves the highest BLEU on Dev set.

[1]https://github.com/pytorch/fairseq

3.2 Experimental Results

Main Results: Our experiments demonstrate the efficiency of our proposed crossed co-attention mechanism, which significantly improves the BLEU scores of machine translation as illustrated in Table 1. Besides, the co-attention mechanism has, by and large, reduced training, validation and test loss from the first training epoch compared with the transformer baselines as shown in Fig. 3,4,5,6. However, since the number of parameters doubles, the epoch time also increases by roughly $60\% \sim 80\%$.

Capability of Model Selection: In addition to the BLEU, loss and time efficiency, we also find that the THM/CCN models demonstrate a better capability of selecting good models with Dev set from all models derived in all training epochs. As is shown in Table 2, for THM/CCN, the models that achieved the highest BLEU on Dev set are also high-ranking on the Test set. In 75% of the cases, THM will select TOP 3 models, and in all cases, it will select TOP 10 models whereas Transformer can only select TOP 10 models in 50% of the cases.

Performance across Languages: We test our proposed method on two language pairs, EN-DE and EN-FI, and the improved BLEU scores and the capability of model selection on both base and big models demonstrate the effectiveness of our proposed method.

4 Related Work

Attention: Multi-head self-attention has demonstrated its capacity in neural transduction models (Vaswani et al., 2017), language model pre-training (Devlin et al., 2018; Radford et al., 2018) and speech synthesis (Yang et al., 2019c). While the novel attention mechanism, eschewing re-

	THM / CCN	Transformer
TOP 1	**25%**	0
TOP 3	**75%**	0
TOP 5	**100%**	0
TOP 10	**100%**	50%

Table 2: This table evaluates if the models selected by the Dev set are also better than others on the test set. Here we provide the percentage of selected models that rank TOP 1, TOP 3, TOP 5 or TOP 10 among all models derived from all training epochs.

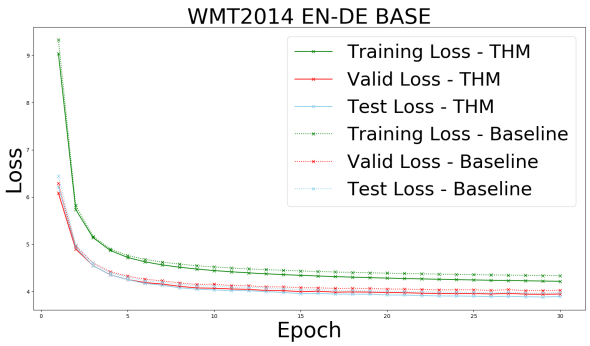


Figure 3: Loss vs Epoch for THM-base and Transformer-base on EN-DE

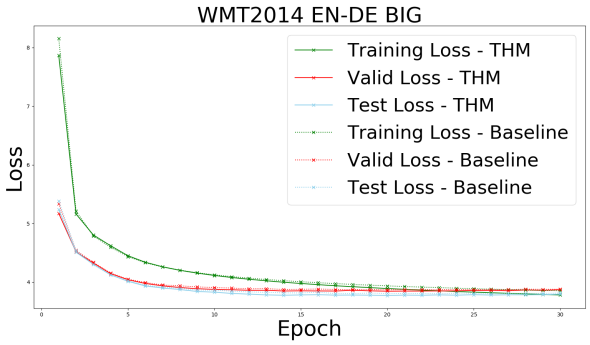


Figure 4: Loss vs Epoch for THM-big and Transformer-big on EN-DE

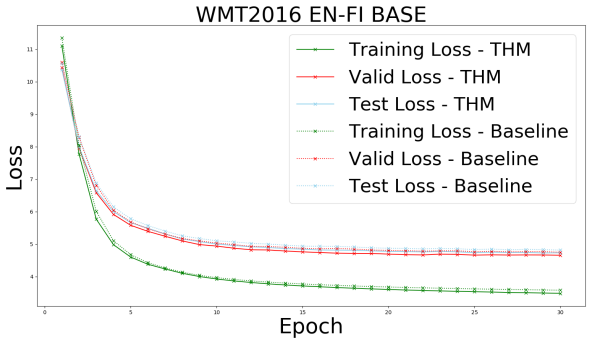


Figure 5: Loss vs Epoch for THM-base and Transformer-base on EN-FI

currence, is famous for modeling global dependencies and considered faster than recurrent lay-

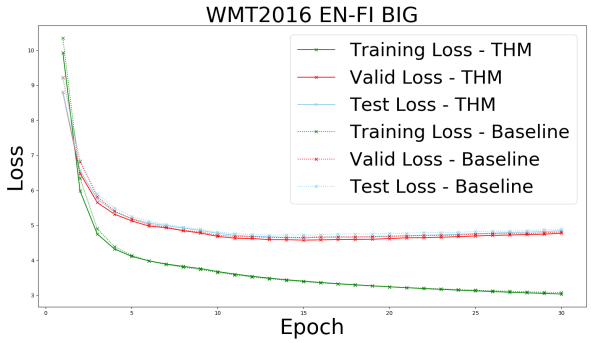


Figure 6: Loss vs Epoch for THM-big and Transformer-big on EN-FI

ers (Vaswani et al., 2017), recent work points out that it may tend to overlook neighboring information (Yang et al., 2019a; Xu et al., 2019). It is found that applying an adaptive attention span could be conducive to character level language modeling tasks (Sukhbaatar et al., 2019). Yang et al. (2018a) propose to model localness for self-attention, which would be conducive to capturing local information by learning a Gaussian bias predicting the region of local attention. Other work indicates that adding convolution layers would ameliorate the aforementioned issue (Yang et al., 2018b, 2019b). Multi-head attention can also be used in multi-modal scenarios when V, K and Q gates take in data from different domains. Helcl et al. (2018) adds an attention layer on top of the encoder-decoder layer with K and V being CNN-extracted image features. Lu et al. (2019) and Tan and Bansal (2019) use co-attention on BERT to learn multi-modal representations for vision-and-language tasks.

Machine Translation: Some recent advances in machine translation aim to find more efficient model architecture based on the Transformer: Hao et al. (2019) add an additional recurrence encoder to model recurrence for Transformer; So et al. (2019) demonstrate the power of neural architecture search and find that the found evolved transformer architecture outperforms human-designed ones; Wu et al. (2019) propose dynamic convolutions that would be more efficient and simpler compared with self-attention. Lample and Conneau (2019) and Liu et al. (2020) investigate how language pretraining can improve machine translation performance. Zhu et al. (2020) studies how to incorporate BERT into machine translation tasks. Other work shows that training on 128 GPUs can significantly boost the experimental results and shorten

the training time (Ott et al., 2018). A novel research direction is semi- or un-supervised machine translation aimed at addressing low-resource languages where parallel data is usually unavailable (Cheng, 2019; Artetxe et al., 2017; Lample et al., 2017).

5 Conclusion and Future Work

We propose Two-Headed Monster (THM) neural structures consisting of two parallel encoders and a decoder connected with each other using co-attention, which is the core module for information flow in THM. First, we formulate the co-attention in a general sense as a non-local operation. Then we design Crossed Co-attention Networks (CCNs) based on our defined co-attention under the umbrella of THM. Our implementation of CCN adopts a specific way of connections of attention gates and improve the machine translation tasks by $0.17 \sim 0.74$ BLEU points and enhance the capability of model selection. However, the time efficiency is reduced since the number of parameters increases.

While we investigate the advantages and disadvantages of our proposed model on machine translation, we still think that the work is at a preliminary stage and our future work will focus on two aspects: (1) adopt input from two different domains and test our model on tasks such as multi-modal machine translation; (2) investigate how different ways of co-attention connections will influence the model performance on different tasks.

Acknowledgments

We thank all anonymous reviewers for their valuable comments and suggestions on our paper. We also thank Dr. Sonit Singh who serves as our mentor in AACL-IJCNLP SRW.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yong Cheng. 2019. Semi-supervised learning for neural machine translation. In *Joint Training for Neural Machine Translation*, pages 25–40. Springer.

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.

Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. 2017. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, 163:90–100.

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*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling recurrence for transformer. *arXiv preprint arXiv:1904.03092*.

Jindřich Helcl, Jindřich Libovickỳ, and Dušan Variš. 2018. Cuni system for the wmt18 multimodal translation task. *arXiv preprint arXiv:1811.04697*.

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.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016a. Hierarchical question-image co-attention for visual question answering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 289–297. Curran Associates, Inc.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016b. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

David R So, Chen Liang, and Quoc V Le. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2309–2318. ACM.

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*.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. *CVPR*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Caiming Xiong, Victor Zhong, and Richard Sochern. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Mingzhou Xu, Derek F Wong, Baosong Yang, Yue Zhang, and Lidia S Chao. 2019. Leveraging local and global patterns for self-attention networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3069–3075.

Baosong Yang, Jian Li, Derek F Wong, Lidia S Chao, Xing Wang, and Zhaopeng Tu. 2019a. Context-aware self-attention networks. *arXiv preprint arXiv:1902.05766*.

Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. 2018a. Modeling localness for self-attention networks. *arXiv preprint arXiv:1810.10182*.

Baosong Yang, Longyue Wang, Derek Wong, Lidia S Chao, and Zhaopeng Tu. 2019b. Convolutional self-attention networks. *arXiv preprint arXiv:1904.03107*.

Shan Yang, Heng Lu, Shiying Kang, Lei Xie, and Dong Yu. 2019c. Enhancing hybrid self-attention structure with relative-position-aware bias for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6910–6914. IEEE.

Zhilin Yang, Jake Zhao, Bhuwan Dhingra, Kaiming He, William W Cohen, Ruslan Salakhutdinov, and Yann LeCun. 2018b. Glomo: Unsupervisedly learned relational graphs as transferable representations. *arXiv preprint arXiv:1806.05662*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 1821–1830.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4995–5004.

Towards a Task-Agnostic Model of Difficulty Estimation for Supervised Learning Tasks

Antonio Laverghetta Jr., Jamshidbek Mirzakhalov, John Licato
Advanced Machine and Human Reasoning (AMHR) Lab
Dept. of Computer Science and Engineering
University of South Florida
Tampa, FL, USA
{alaverghett,mirzakhalov,licato}@usf.edu

Abstract

Curriculum learning, a training strategy where training data are ordered based on their difficulty, has been shown to improve performance and reduce training time on various NLP tasks. While much work over the years has developed novel approaches for generating curricula, these strategies are typically only suited for the task they were designed for. This work explores developing a task-agnostic model for problem difficulty and applying it to the Stanford Natural Language Inference (SNLI) dataset. Using the human responses that come with the dev set of SNLI, we train both regression and classification models to predict how many annotators will answer a question correctly and then project the difficulty estimates onto the full SNLI train set to create the curriculum. We argue that our curriculum is effectively capturing difficulty for this task through various analyses of both the model and the predicted difficulty scores.

1 Introduction

Recent advances on natural language processing (NLP) benchmarks have been driven by increasingly sophisticated language models, which are pretrained on enormous amounts of data before use. Refinements of this process has led to increasingly powerful language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and more recently T5 (Raffel et al., 2019). Such models are quickly saturating even new tasks that have undergone a rigorous adversarial filtering process (Zellers et al., 2018). However, these downstream performance improvements also require more computational resources and data to train the models, which is not always feasible. Curriculum learning (Elman, 1993), a strategy where the model is trained on easier examples before harder ones, has recently been shown to improve performance and reduce training time on a variety of NLP tasks, especially machine translation (Liu et al., 2020; Wang et al., 2020; Zhou et al., 2020; Xu et al., 2020).

The success of this research shows that the order in which training data is presented to a model is important, but how to best apply curriculum learning broadly to NLP and how to design effective measures of difficulty to create curricula remain relatively unexplored. Prior approaches either use a hand-crafted measure of difficulty that works well for a particular task or design an architecture that automatically creates the curriculum during training. In either case, the curriculum is formed using some information-theoretic measure of difficulty (semantic distance, feedback from a separate network, etc.), and it is difficult to interpret why they work well for some tasks and not others. In a practical sense, it is seldom clear how to apply a previously investigated curricula directly to another task.

In this paper, we explore how to address these shortcomings by creating what we call a *task-agnostic* model of difficulty, which we argue can, in principle, be applied to any supervised learning task. We use this model to investigate what makes a good difficulty measure for curricula beyond how it affects downstream performance to better explain why one curriculum should be preferred over another. We use the well-known SNLI dataset for our experiments (Bowman et al., 2015), which provides a high-quality data source for the natural language inference (NLI) task. Given a premise sentence, the goal is to predict whether a hypothesis sentence is entailed by the premise, contradicted by it, or is neither entailed nor contradicted by it (neutral). Unlike many NLI datasets, SNLI includes in its dev set the responses from each crowd-sourced worker who attempted to answer the question, giving us a set of 10,000 annotated questions to create a difficulty model. We release all code related to our

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 16–23

experiments on GitHub.[1]

2 Related Work

The idea of applying concepts from developmental psychology to artificial intelligence traces as far back as Turing (1950). Although much work over the years has explored variants of this idea, mostly under the subject title "Developmental AI" (Elman, 1993; Shultz, 2003; Shultz and Sirois, 2008; Asada et al., 2009; Bruce, 2010; Guerin, 2011; Licato and Bringsjord, 2016; Vernon et al., 2016), only recently has the target task been defined as that of determining how to organize training data of machine learning models so that they can benefit from the kind of scaffolded learning that a human tutor might provide to a child.

Bengio et al. (2009) demonstrated that following a curriculum can provide a more optimal solution during gradient descent. Other early work includes Krueger and Dayan (2009), which experimented with shaping tasks (Skinner, 1938) learned by a LSTM model. These early results suggest that, for many learning tasks, estimating the difficulty of problems in a training set may indeed be simpler than training on random permutations of the dataset, but determining an effective measure of difficulty remains far from trivial. Previous work has addressed this problem in two fundamentally different ways: by using a separate model from that used on the main task itself to predict difficulty (Kumar et al., 2010; Graves et al., 2017; Jiang et al., 2018; Matiisen et al., 2019; Shen and Feng, 2020); or using a manually designed measure of difficulty not obtained through learning (Platanios et al., 2019; Liu et al., 2020; Xu et al., 2020).

A crucial question which recent AI work has not sufficiently addressed is how humans design curricula, which was stressed as essential to understand by early research in this area (Bengio et al., 2009; Taylor, 2009). For example, much has been said about the use of *scaffolding* in child education, in which a student is provided examples by a more experienced (typically older) tutor, such that those examples are neither "too easy" nor "too hard" given their current skill level (a space called the zone of proximal development) (Vygotsky, 1978). In reinforcement learning, there has been fruitful research that applies these ideas by using human feedback to manually create curricula (Stanley et al., 2005; Thomaz and Breazeal, 2006; Suay and Chernova, 2011; Loftin et al., 2016). We seek to apply these ideas in the context of supervised learning.

[1] https://github.com/AMHRLab/supervised-CL

3 Difficulty Model Experiments

Using the SNLI dev set's annotation data, our goal is to train a model that can estimate the difficulty of the questions in the train set, hence creating a measure of difficulty. Given the prior success of human-in-the-loop training, we believe this provides a natural way to estimate difficulty for any task where human annotation data can be collected. Each question in the dev set contains 5 different labels assigned by each annotator, along with the gold label deemed correct for the question. The model's specific objective is to predict how many of 5 possible respondents predicted the gold label correctly, and this percentage is the approximate difficulty.

The most intuitive solution to this problem is to treat it as a regression task, and we experiment with several regression models (Section 3.1). However, because the notion of difficulty among humans is a somewhat fuzzy concept, it's unclear to us if a floating-point difficulty score will be meaningful up to arbitrary precision. Therefore, we also experiment with framing the objective as a classification task by mapping the 5 possible percentages into labels and then training various classification models (Section 3.2).

Using features from prior work, we create 5 different feature sets as inputs to both sets of models. The first is sentence length, which has demonstrated to be useful to measure difficulty in machine translation tasks (Platanios et al., 2019). We adapt this to work for SNLI by taking the sentence length of the premise and hypothesis. Second is the RoBERTa sentence embeddings of the premise and the hypothesis. We feed the raw embeddings in by concatenating the entire hypothesis vector to the premise vector, and also by stacking them pairwise such that dimension i of the premise vector is paired with dimension i of the hypothesis vector in the final representation. We refer to this as the flattened embedding. Finally, we use cosine similarity to measure the semantic distance between the input sequences. Table 1 summarizes the various feature sets we have used.

3.1 Regression

Table 2 shows the Spearman's correlation (Spearman, 1904) measured for all regression models.

Feature set
1: cosine similarity + sentence length
2: flattened embeddings
3: concatenated embeddings
4: concatenated embeddings + sentence length
5: concatenated embeddings + cosine similarity

Table 1: Feature sets used for experiments

We experiment with various linear, non-linear, and neural models and perform a grid search over each model's hyperparameters to ensure we have achieved an optimal correlation in each case. For each trial, we shuffle the dataset before training. We use the scikit-learn[2] implementation of all the non-neural models, and scipy[3] to measure correlation. A decision tree using feature set 1 with a maximum depth of 40 achieves the best correlation of all models, with an average observed correlation of 84%. Support vector regression using feature set 2 with C=1 also achieves a strong mean correlation of about 74%. To verify that results for these top-performing models achieve statistically significant results, we also perform 10-fold cross-validation experiments for each of them. The decision tree model achieves a mean correlation of 0.895 across all folds of cross-validation, and the SVR model 0.749.

We conduct two types of neural regression experiments: finetuning a pretrained RoBERTa model and training a multi-layer deep neural network (DNN) from scratch. For the DNN model, we experiment with a simple feed-forward neural network with {2,3,4,5} hidden layers and dimension size of {512, 1024, 2048}. As common with many large language models used for relatively small datasets, RoBERTa quickly overfits to the data and was unable to generalize to the test set. DNN models, on the other hand, failed to capture any significant relationship between the feature sets and the difficulty estimation, as they performed the worst out of all models. We ran an extensive hyperparameter search over the learning rates, batch sizes, the number of layers, and the number of epochs for both RoBERTa and DNN experiments. We use the simpletransformers[4] implementation of RoBERTa and PyTorch[5] for the DNN experiments.

[2]https://scikit-learn.org/stable/index.html
[3]https://www.scipy.org
[4]https://github.com/ThilinaRajapakse/simpletransformers
[5]https://pytorch.org/

Model	Mean	Min	Max
SVR	0.737	0.714	0.755
Decision Tree	0.84	0.801	0.891
Linear	0.299	0.277	0.334
KNN	0.361	0.333	0.383
DNN	0.151	0.09	0.166
RoBERTa-large	0.264	0.215	0.35

Table 2: Spearman's correlation for all regression models.

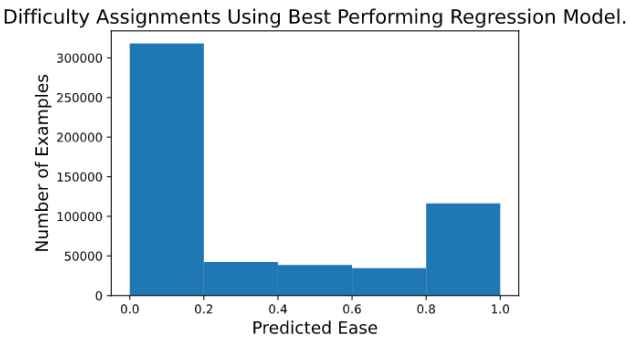


Figure 1: Histogram of difficulty scores for SNLI using decision tree regression, which was the best performing regression model.

3.2 Classification

To perform classification, we map the five possible regression scores onto discrete labels. However, after mapping, we found that the resulting classes were imbalanced. For instance, the number of examples where all annotators correctly predicted the gold label was 10 times more than the examples where no one predicted the label correctly. To address this issue, we oversampled the minority classes using imblearn[6] before training our models. Table 3 shows performance statistics for each model. We again perform a grid search to optimize the feature set choice and hyperparameters for each model. Similar to regression, a decision tree classifier with feature set 1 and a depth of 40 achieves the best performance by a large margin. We found that using either Gini impurity (Breiman et al., 1984) or entropy to measure the quality of the splits achieved about the same performance. We additionally evaluate the same decision tree model using 10 fold cross-validation and achieve the same mean accuracy.

3.3 Analysis of Models

The previous results demonstrate that we can create a model that significantly correlates with the actual

[6]https://github.com/scikit-learn-contrib/imbalanced-learn

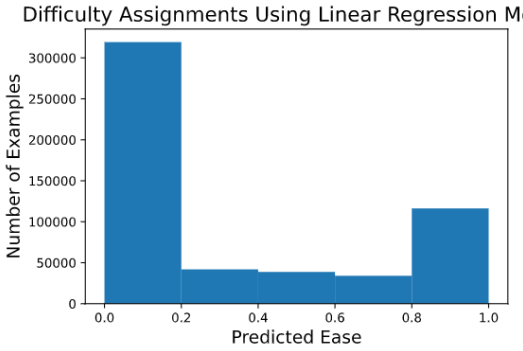


Figure 2: Histogram of difficulty scores for SNLI using the linear regression model.

Model	Mean	Min	Max
SVC	0.304	0.291	0.312
Decision Tree	0.834	0.821	0.854

Table 3: Accuracy for all classification models.

human responses to the SNLI dev set. We now perform additional quantitative and qualitative analyses to gain a better understanding of how well these models actually predict difficulty, and by extension how good any curriculum resulting from it will be. Previous work has tended to evaluate the quality of the curriculum primarily in terms of downstream performance; a curriculum is considered good if using it improves performance or reduces training time. However, many factors will ultimately affect model performance beyond the use of any curricula (hyperparameters, training strategy, etc). In the context of human education, it has long been appreciated that curricula are complex and cannot be evaluated using any single metric (Macdonald, 1971; Kliebard, 1989; Kelly, 2009; Pinar, 2012). Therefore, we believe it is important to attempt to evaluate the quality of our model separate from its application to any task.

Using the best regression and classification models, we predict the difficulty of each problem in the SNLI train set. We additionally do the same using the linear regression model, which achieves only weak correlation, as a point of comparison. Figures 1 and 2 show the distributions of difficulty assignments for the best performing regression model (the decision tree) and the linear regression model, respectively. Higher scores represent easier problems in both figures. Table 4 shows examples rated with both maximum and minimum difficulty from both models. Interestingly, both models predict the same general trend in the difficulty distributions. However, as Table 4 shows there are cases where the models make substantially different difficulty predictions about the same question. The question in the first row of the easy column is rated with minimum difficulty by the best performing model, whereas the same question is rated with maximum difficulty by the linear model. To better understand the differences between each model's predictions, we computed the difference between the predicted difficulty scores for each question in the train set and normalized them by taking the absolute value. About 27% of the questions have the same prediction from both models (difference of estimated score is 0), and about 15.8% have polar opposite predictions (the difference is 1). Furthermore, the mean difference between the scores from both models is 0.41, suggesting that they make substantially different predictions even though they report the same overall distribution.

Figure 3 shows the distributions of difficulty scores from the classification models. We similarly observed wide variation in the assigned difficulty, however in this case most problems are assigned as being very easy. This is probably the most accurate model, given that in the majority of cases the annotators were able to correctly predict the gold label (Figure 4).

These results give us some confidence that our curriculum may lead to a performance gain on SNLI, given a suitable model to train it with. Designing a curriculum for both humans and machine learning models requires that there be some difference in difficulty among the questions in the task, otherwise any random permutation would be essentially the same curriculum. That all models are reporting variance in the difficulty distribution indicates that there may be enough difference in difficulty for curriculum learning to help in this case. We hypothesize that the linear model, despite having only very weak correlation with the human responses, is still able to capture the high level structure of the difficulty distribution, which would explain why the two histograms are identical. However, when we examine the predictions at a finer level it becomes obvious the models are making fundamentally different regression decisions.

4 Conclusion and Future Work

In humans, teaching, especially according to a curriculum, reduces the difficulty of learning complex skills by providing a simpler path for learners to

Easy	Hard
Decision Tree Model: *Premise:* A white dog runs through a field. *Hypothesis:* A white dog is running back to his master. neutral	*Premise:* A man is surfing in a bodysuit in beautiful blue water. *Hypothesis:* On the beautiful blue water there is a man in a bodysuit surfing. entailment
Premise: A group of workers are posing for a picture. *Hypothesis:* A group of workers are playing baseball. contradiction	*Premise:* A man is kneeling in the top step while many people are behind him sitting in chairs. *Hypothesis:* A man sleeps comfortably at home. contradiction
Linear Regression Model: *Premise:* A small child is running on the shore of the beach surrounded by birds. *Hypothesis:* Birds surround a child looking for food. neutral	*Premise:* A white dog is running through a field. *Hypothesis:* A white dog is running back to its master. neutral
Premise: A group of people are hiking in the forest. *Hypothesis:* People are hiking in the forest. entailment	*Premise:* An old man is standing before a crowd to perform a feat. *Hypothesis:* An oldtimer about to perform a group of people. entailment

Table 4: SNLI questions predicted as being very easy and as very challenging by both the decision tree model and the linear regression model. The first two rows are predictions from decision tree model and the last two rows are from the linear regression model. The left column are questions rated as **Easy** by the respective model, while the right column are questions rated as **Hard**. Gold labels are shown in blue

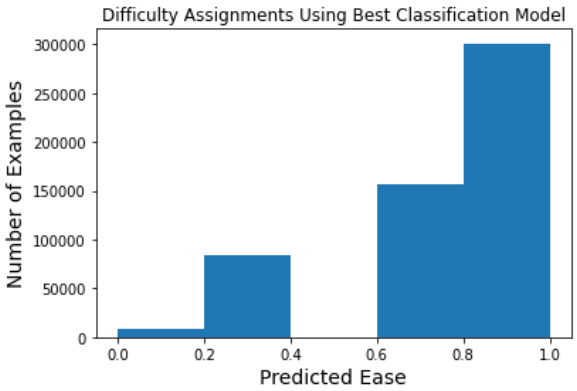


Figure 3: Histogram of difficulty scores for SNLI using the best classification model.

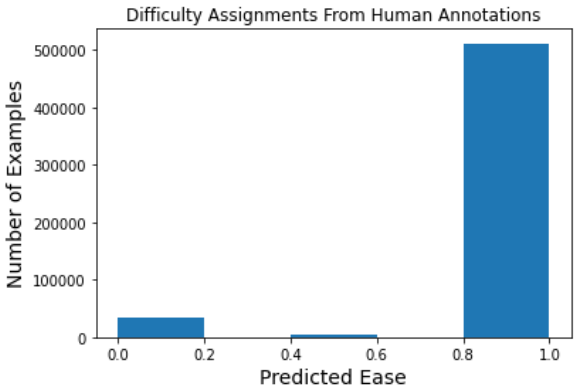


Figure 4: Histogram of difficulty scores for SNLI using the human annotations.

follow. While much work has explored various applications of curriculum learning over the years, the larger problem of what makes a good difficulty measure remains an important research question. In this work, we have presented initial studies on designing a task-agnostic measure of difficulty for curricula. While for practical reasons, we chose to focus specifically on an NLP task, in principle our approach can be extended to any supervised learning task where it is possible to gather human annotations. Our analysis suggests that the resulting curriculum is suitable for use in a curriculum learning algorithm, firstly because it correlates well with human difficulty estimates, and secondly because it displays wide differences in the estimated difficulties across all the problems in the train set.

An important question to answer next is how to actually apply the curriculum to the training of a model. While this may appear trivial, the wide array of factors that affect downstream performance means that it cannot be taken lightly. The model choice is an especially important factor; using a re-

cent model which already does very well on SNLI (Zhang et al., 2020) is unlikely to have any effect because performance on the task has saturated at this point. However, if we apply a model which is too simplistic, for instance, the classic n-gram language model, then it's unlikely that even the best curriculum will help such a model to learn a task which is probably far too challenging for it. A study examining how our curriculum affects downstream performance for models ranging from the state-of-the-art to the baseline would help answer this question.

Another important factor is the sampling protocol. Feeding examples to the model in order of difficulty, with easier ones preceding harder ones, is the most straightforward solution. However, there is no guarantee that this is the optimal way to feed examples into the model. In fact, there have been cases where feeding examples in *reverse* order, with hard examples preceding easier ones, has lead to optimal performance improvement (McCann et al., 2018). Future work will also investigate the effect of various sampling strategies.

Finally, something which should be considered is the quality of the gold labels which our models are trying to predict. Since NLI is an inherently ambiguous task, determining the ground truth of a given NLI question is challenging. Recent work by Nie et al. (2020b) has shown that many of the gold labels for both SNLI and the MNLI dataset (Williams et al., 2018), which also includes the responses of individual annotators, will change when more annotators are used with stricter quality control protocols, though they only found this to be true when agreement on the correct label was low to begin with. Our choice for using SNLI specifically is that, unlike more recent datasets (Bhagavatula et al., 2020; Nie et al., 2020a), it also includes the label predicted by each individual annotator and not just the final gold label. We might be able to account for the fuzziness of the ground truth in the final difficulty prediction of our model. For instance, if the agreement of the gold label for a question is low, we could take this into consideration and predict the question as being even more difficult. This avenue will be explored as a way to improve the quality of the difficulty model, using both SNLI and similar datasets, such as MNLI, as part of a more comprehensive study.

Acknowledgements

This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-17-1-0191 and FA9550-18-1-0052. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

References

Minoru Asada, Koh Hosoda, Yasuo Kuniyoshi, Hiroshi Ishiguro, Toshio Inui, Yuichiro Yoshikawa, Masaki Ogino, and Chisato Yoshida. 2009. Cognitive Developmental Robotics: A Survey. *IEEE Transactions on Autonomous Mental Development*, 1(1):12–34.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. 2020. Abductive Commonsense Reasoning. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.

David Bruce. 2010. Cognitive Architecture and Testbed for a Developmental AI System. Undergraduate Honours Thesis.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org.

Frank Guerin. 2011. Learning Like a Baby: A Survey of Artificial Intelligence Approaches. *The Knowledge Engineering Review*, 26(2):209–236.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *International Conference on Machine Learning*, pages 2304–2313.

Albert Victor Kelly. 2009. *The curriculum: Theory and practice*. Sage.

Herbert M Kliebard. 1989. Problems of definition in curriculum. *Journal of Curriculum and Supervision*, 5(1):1–5.

Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

John Licato and Selmer Bringsjord. 2016. A Physically Realistic, General-Purpose Simulation Environment for Developmental AI Systems. In *Proceedings of the ECAI 2016 Workshop on Evaluating General-Purpose AI (EGPAI 2016)*.

Xuebo Liu, Houtim Lai, Derek F Wong, and Lidia S Chao. 2020. Norm-Based Curriculum Learning for Neural Machine Translation. *arXiv preprint arXiv:2006.02014*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Robert Loftin, Bei Peng, James Macglashan, Michael L. Littman, Matthew E. Taylor, Jeff Huang, and David L. Roberts. 2016. Learning behaviors via human-delivered discrete feedback: Modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59.

James B Macdonald. 1971. Curriculum theory. *The Journal of Educational Research*, 64(5):196–200.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2019. Teacher-Student Curriculum Learning. *IEEE Transactions on Neural Networks*, pages 1–9.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The Natural Language Decathlon: Multitask Learning as Question Answering. *arXiv preprint arXiv:1806.08730*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020a. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *ACL 2020: 58th annual meeting of the Association for Computational Linguistics*, pages 4885–4901.

Yixin Nie, Xiang Zhou, and Mohit Bansal. 2020b. What Can We Learn from Collective Human Opinions on Natural Language Inference Data. *arXiv preprint arXiv:2010.03532*.

William F Pinar. 2012. *What is curriculum theory?* Routledge.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based Curriculum Learning for Neural Machine Translation. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1162–1172.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint arXiv:1910.10683*.

Lei Shen and Yang Feng. 2020. CDL: Curriculum Dual Learning for Emotion-Controllable Response Generation. In *ACL 2020: 58th annual meeting of the Association for Computational Linguistics*, pages 556–566.

Thomas R. Shultz. 2003. *Computational Developmental Psychology*. The MIT Press, Cambridge, Massachusetts.

Thomas R. Shultz and Sylvain Sirois. 2008. Computational Models of Developmental Psychology. In Ron Sun, editor, *The Cambridge Handbook of Computational Psychology*, chapter 16, pages 451–476. Cambridge Univ Press, New York, New York, USA.

BF Skinner. 1938. The behavior of organisms: an experimental analysis.

Charles Spearman. 1904. The proof and measurement of association between two things. *American journal of Psychology*, 15(1):72–101.

Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. 2005. Evolving neural network agents in the NERO video game. *Proceedings of the IEEE*, pages 182–189.

Halit Bener Suay and Sonia Chernova. 2011. Effect of human guidance and state space size on Interactive Reinforcement Learning. In *2011 RO-MAN*, pages 1–6.

Matthew E. Taylor. 2009. Assisting Transfer-Enabled Machine Learning Algorithms: Leveraging Human Knowledge for Curriculum Design. In *AAAI Spring Symposium: Agents that Learn from Human Teachers*, pages 141–143.

Andrea L. Thomaz and Cynthia Breazeal. 2006. Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In *AAAI'06 Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, pages 1000–1005.

Alan M. Turing. 1950. Computing Machinery and Intelligence. *Mind*, pages 433–460.

David Vernon, Claes van Hofsten, and Luciano Fadiga. 2016. Desiderata for Developmental Cognitive Architectures. *Biologically Inspired Cognitive Architectures*, 18:116–127.

Lev Vygotsky. 1978. *Mind in Society: The Development of Higher Psychological Processes*.

Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. 2020. Learning a Multi-Domain Curriculum for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7711–7723.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. 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 1112–1122. Association for Computational Linguistics.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum Learning for Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP 2018: 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuiliang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware BERT for Language Understanding. *AAAI 2020 : The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 34(5):9628–9635.

Yikai Zhou, Baosong Yang, Derek F Wong, Yu Wan, and Lidia S Chao. 2020. Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6934–6944.

A Siamese CNN Architecture for Learning Chinese Sentence Similarity

Haoxiang Shi
Waseda University
Tokyo, Japan
hollis.shi@toki.waseda.jp

Cen Wang
KDDI Research, Inc.
Saitama-ken, Japan
ce-wang@kddi-research.jp

Tetsuya Sakai
Waseda University
Tokyo, Japan
tetsuyasakai@acm.org

Abstract

This paper presents a deep neural architecture which applies the Siamese Convolutional Neural Network sharing model parameters for learning a semantic similarity metric between two sentences. In addition, two different similarity metrics (i.e., the Cosine Similarity and Manhattan similarity) are compared based on this architecture. Our experiments in binary similarity classification for Chinese sentence pairs show that the proposed Siamese convolutional architecture with Manhattan similarity outperforms the baselines (i.e., the Siamese Long Short-Term Memory architecture and the Siamese Bidirectional Long Short-Term Memory architecture) by 8.7 points in accuracy.

1 Introduction

Measuing the similarity between words, sentences, paragraphs and documents is an important component in various tasks such as information retrieval, document clustering, word-sense disambiguation, automatic essay scoring, short answer grading, machine translation and text summarization. Traditional sentence similarity measurement is based on the edit distance, Jaccard index, and the bag-of-words models such as TF-IDF. These methods of learning sentence similarity are in fact based on the word level, which may not be sufficient. For example, there are two Chinese sentences as shown in Figure 1. The corresponding English translations are "How to buy LCD TVs." and "What kind of LCD TVs is good?". From the word level (i.e., character level in Chinese), the two sentences look the same, but they have totally different meaning at the sentence level. That is, we need sentence-level methods to capture the semantics of the sentences for sentence similarity measurement.

With the rapid development of machine learning, using neural network to learn representations of sentence-level meanings has been widely verified to be effective. The beginning of using neural network to learn sentence-level representations may be the Word2Vec from Google (Mikolov et al., 2013), which used a shallow structure to learn the vector-based representations of sentence level. However, using one neural architecture to learn two sentences in two steps may cause inconsistent representations. Hence, Siamese structures, which can learn two sentences at a time, are attractive alternatives. The Siamese architecture that can achieve state-of-the-art accuracy results in learning English sentence similarity is a Bidirectional Long Short-Term Memory (Bi-LSTM) based Siamese recurrent architecture (Neculoiu et al., 2016).

In our preliminary study, we tested the effectiveness of a Siamese recurrent architecture for learning Chinese sentence similarities. However, this did not perform as well as what is reported in (Neculoiu et al., 2016). Therefore, we borrowed a Siamese convolutional architecure from the image processing field (Koch et al., 2015) to implement Chinese sentence similarity learning. The results in binary similarity classification for Chinese sentence pairs show that Siamese convolutional architecture outperforms the Siamese recurrent architecture in learning accuracy. In addition, we consider two similarity metrics in the Siamese convolutional architecture, namely, the Manhattan similarity and the Cosine similarity. The results show that the Siamese convolutional architecture plus the Manhattan similarity performs better than other baselines for learning the similarity between two Chinese sentences. Our contributions are as follows: (1) we verified that Siamese convolutional architecture is effective in learning Chinese sentence semantic similarity; (2) we verified that the Manhattan similarity can achieve better performance than other similarity metrics regardless of the learning architectures.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 24–29

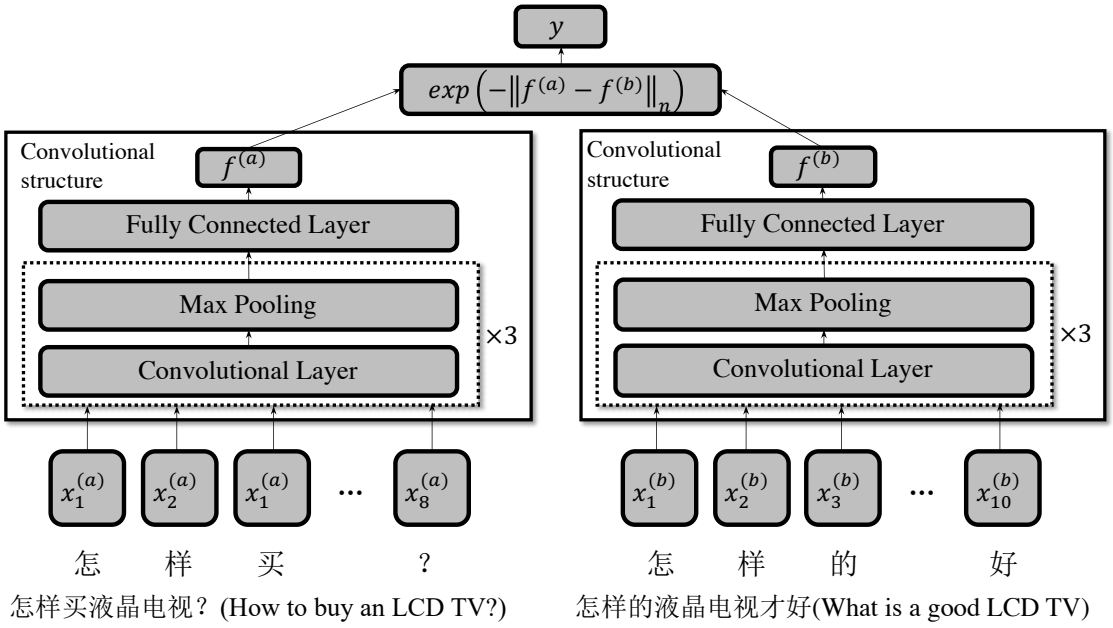


Figure 1: Siamese convolutional architecture.

2 Related Work

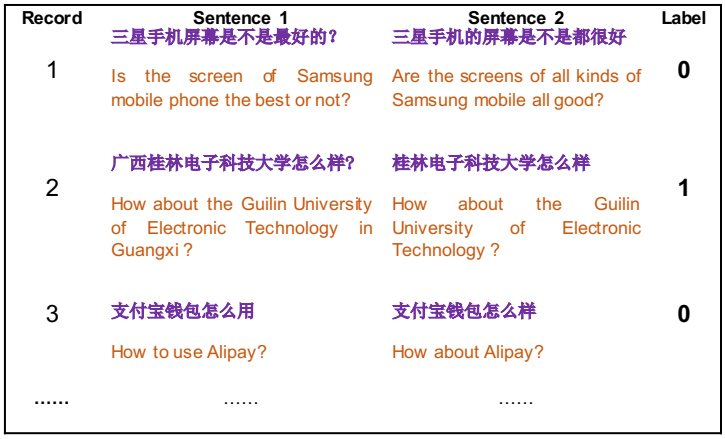
The Siamese network (Bromley et al., 1993) is firstly proposed for non-linear metric learning with similarity information. It naturally learns representations that embody the invariance and selectivity desiderata through explicit information about similarity between pairs of objects. The Siamese architecture has since been widely used in vision applications. Specifically, the Siamese convolutional networks were used to learn complex similarity metrics for face verification (Chopra et al., 2005) and dimensionality reduction on image features (Hadsell et al., 2006). While in the natural language processing (NLP) field, the Convolutional Neural Network (CNN) has attracted more attentions since the successes in using CNN to do the traditional NLP tasks (Collobert et al., 1993), and the availability of high-quality semantic word representations has been verified when using the CNN (Mikolov et al., 2013).

Recently, CNNs have been applied to matching sentences (Hu et al., 2014). Although the work (Hu et al., 2014) has used the CNN to learn representations of two sentences, this is not a Siamese CNN architecture. Following this, the Siamese Long Short-Term Memory (LSTM) architecture was proposed for sentence similarity task using token level embedding (Mueller and Thyagarajan, 2016). Subsequently, a Siamese Bi-LSTM structure was proposed in order to improve the result of sentence similarity (Neculoiu et al., 2016). A Siamese CNN combines Bi-LSTM structure has been proposed for learning sentence similarity (Pontes et al., 2018). However, this architecture achieves lower accuracy than the independent Bi-LSTM structure. Also, the paper (Pontes et al., 2018) did not give any comparisons between Siamese CNN architecture and Siamese Bi-LSTM architecture. Later, Siamese LSTM and Siamese Bi-LSTM were compared based on an English dataset (Ranasinghe et al., 2019).

3 Siamese Convolutional Architecture

The proposed Siamese convolutional architecture is depicted in Figure 1. In the architecture, there are two exactly alike convolutional structures that are used. The inputs of each convolutional structure are the character-level embeddings of a sentence, and the outputs of each convolutional structure are the sentence level representations. Then, a similarity metric is used to compare the outputs of the two convolutional structures. The calculated similarity is the final output of the Siamese convolutional architecture.

Within each convolutional architecture, there are one fully connected layer after three repeated convolutional layers and max pooling layers. We have also tested the six repeated structure, but the accuracy did not show a significant improvement. The kernel size of each convolutional layer is different. A higher convolutional layer is equipped



Record	Sentence 1	Sentence 2	Label
1	三星手机屏幕是不是最好的？ Is the screen of Samsung mobile phone the best or not?	三星手机的屏幕是不是都很好 Are the screens of all kinds of Samsung mobile all good?	0
2	广西桂林电子科技大学怎么样? How about the Guilin University of Electronic Technology in Guangxi ?	桂林电子科技大学怎么样 How about the Guilin University of Electronic Technology ?	1
3	支付宝钱包怎么用 How to use Alipay?	支付宝钱包怎么样 How about Alipay?	0
.....	.....	.....	

Figure 2: The format of the LCQMC.

with a larger kernel size. The fully connected layer then reduces the dimension of the learned representations from pooling layer. The learned output vector from the fully connected layer will be used to calculate the similarity then.

The similarity depicted in Figure 1 is the exponential negative norm of two learned representation vectors, which is defined as:

$$sim_{Man} = \exp\left(-\left\|f^{(a)} - f^{(b)}\right\|_n\right) \quad (1)$$

where in equation (1) $f^{(a)}$ and $f^{(b)}$ are the representations of the two sentences from the two convolutional structures. If $n = 1$, the similarity is the Manhattan distance-based similarity or the Manhattan similarity for short. If $n = 2$, the similarity is then the Euclidean distance-based similarity or the Euclidean similarity for short. We have also tested the performance of the Siamese network with Euclidean similarity. The accuracy is around 50%, which means the Euclidean similarity does not work well with the Siamese architecture. Therefore, this result is not shown in Section 4. The similarity can also be replaced by the Cosine similarity.

$$sim_{Cos} = \frac{(f^{(a)} \cdot f^{(b)})}{\left\|f^{(a)}\right\| \cdot \left\|f^{(b)}\right\|} \quad (2)$$

After calculating the similarity, we then use the mean-square error (MSE) of the similarity and the label as the loss function. The gradients of the loss will be fed back to both convolutional structures. In this way, the two convolutional structures will share the same parameters, and then they can learn the representations of the two sentences with the same distribution. Based on a threshold of the similarity, we can then evaluate the accuracy after learning.

4 Experiments

Our experiments are the binary similarity classification tasks for Chinese sentence pairs. Although obtaining a Chinese sentence similarity dataset is difficult, we found a dataset named LCQMC with even distribution of the labels (i.e., similar sentence pairs and dissimilarity sentence pairs occupy 50% and 50% of all dataset respectively) from Baidu. The format of the dataset is shown in Figure 2. Punctuations of some sentences are omitted in the original data. This dataset consists 283,000 data records. We have chosen 250,000 data records as the training data, and 12,500 data records as the test data. A data record is like <sentence 1, sentence 2, similarity> (i.e., 1 represents that the two sentences are similar and 0 represents that two sentences are dissimilar). We also used the English dataset PAWS-X (Yang et al., 2019) to train and test the different saimese architecture as the comparisons. In the PAWS-X, the data format is the same with LCQMC, and the task is also to learn the semantic similarity between two sentences. We used 49,401 data records in PAWS-X to train models and 2,000 to test.

From the aforementioned related works, (Neculoiu et al., 2016) and (Ranasinghe et al., 2019), we have chosen two baselines: the Siamese Bi-LSTM architecture and the Siamese LSTM architecture. Moreover, we also evaluated the two baslines and the Siamese convolutional architecture with two different loss functions (i.e., the Manhattan simi-

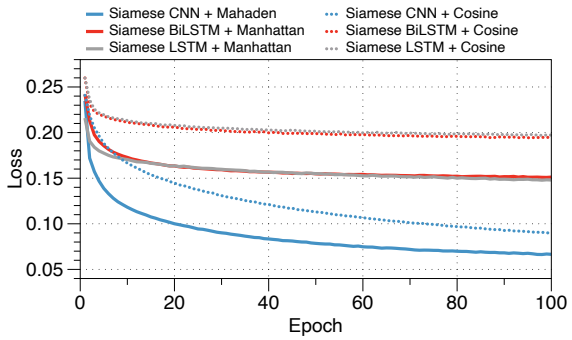


Figure 3: The convergences and the losses of the Siamese convolutional architectures and the baselines.

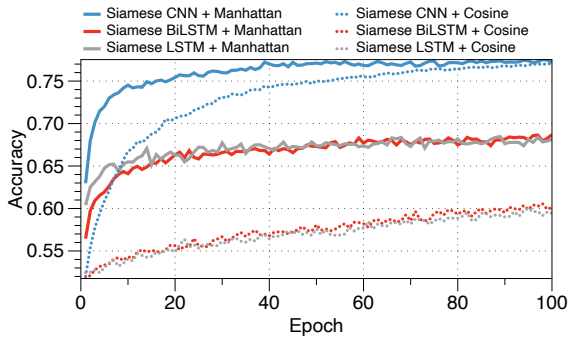


Figure 4: The accuracies of the Siamese convolutional architectures and the baselines.

larity based MSE and the Cosine distance based MSE).

As for the Siamese convolutional architecture, the kernel sizes of the three repeated convolutional layers are set as 3, 4 and 5. We ran a total of 100 epochs and the batch size of each epoch is 128. The Adam optimizer is used. During the optimization, we set the learning rate to be 0.001. Following previous work (Neculoiu et al., 2016), we used accuracy as the evaluation metric. We then set the similarity threshold as 0.5. That is to say, if the calculated similarity is more than 0.5, the prediction is that the two sentences are similar. Conversely, the similarity less than 0.5 is decided as dissimilar.If the similarity is exactly 0.5, the result is excluded for calculating accuracy.

In Figure 3, we compared the convergence speeds and losses of all combinations of the Siamese architectures and the two loss functions. The lines in different colors represent different Siamese architectures. The full lines are the losses using the Manhattan similarity, and the dotted lines are the losses using the Cosine similarity. It can be observed that no matter what kind of the Siamese architecture is used, the Manhattan similarity based Siamese architectures converge fast. As for the loss, the Siamese convolutional architectures always achieve lower losses than the baselines. In the Siamese convolutional architectures, the Manhattan similarity based Siamese architecture always gets a lower loss. As a result, the Siamese convolutional architecture with the Manhattan similarity metric achieves the lowest loss. Regardless of the choice the similarity metric, the losses of the Siamese LSTM architecture and the Siamese Bi-LSTM architecture are similar.

Next, we evaluated the accuracy of all the combinations of the Siamese architectures and the two loss functions. The representation formats of different combinations are the same with Figure 3. As shown in Figure 4, it can be seen that the Siamese convolutional architectures always achieve higher accuracy. In the Siamese convolutional architectures, the one with the Manhattan similarity metric always achieves higher accuracy. In summary, the Siamese convolutional architecture with the Manhattan similarity metric can obtain the highest accuracy. The performances of the two baselines are not substantially different regardless of the similarity metric. The Siamese Bi-LSTM architecture shows a slight improvement of the accuracy comparing to the Siamese LSTM architecture.

We listed all the experimental results in the Table 1, including using LCQMC dataset and PAWS-X dataset. It can be observed that when using LCQMC dataset, both Siamese convolutional architecture with the Manhattan similarity metric and with the Cosine similarity metric outperform the Siamese Bi-LSTM architecture and the Siamese LSTM architecture. Specifically, our Siamese convolutional architecture with Manhattan similarity metric outperforms the Siamese Bi-LSTM architecture with Manhattan similarity by 8.67 points and the Siamese LSTM architecture by 8.68 points respectively. In addition, our Siamese convolutional architecture with Cosine similarity metric also outperforms the Siamese Bi-LSTM architecture with the same similarity metric by 16.50 points and the Siamese LSTM architecture with the same similarity metric by 17.03 points respectively. However, when using PAWS-X, the English dataset, the Siamese LSTM architecture and Siamese Bi-LSTM architecture outperform Siamese convolutional architecture. In the experiments of learning English dataset, to improve the performance of the Siamese LSTM and

Dataset	Architecture	Manhattan Similarity	Cosine Similarity
LCQMC	Siamese convolutional architecture	**77.31**	**77.05**
	Siamese Bi-LSTM architecture	68.64	60.55
	Siamese LSTM architecture	68.63	60.02
PAWS-X	Siamese convolutional architecture	57.80	56.41
	Siamese Bi-LSTM architecture	67.75	**69.20**
	Siamese LSTM architecture	**68.14**	67.45

Table 1: Accuracy comparison for different architectures with Manhattan and Cosine similarities.

Bi-LSTM architectures, introducing Glove (Pennington et al., 2014) may be effective. The performance discrepancy of the Siamese convolutional architecture between Chinese and English may because that part of the CNN can do character-level encoding for Chinese. This is also why in some Chinese language tasks such as (Dai and Cai, 2017) and (Su and Lee, 2017), a CNN-based character-level encoder is added before the word-level or sentence-level encoding

5 Conclusion and Future Work

We proposed a Siamese convolutional architecture for Chinese sentence similarity learning. The experimental results have verified that the Siamese convolutional architecture outperforms the Siamese Bi-LSTM architecture and the Siamese LSTM architecture in terms of accuracy. Moreover, with the proposed architecture, we learned that for Chinese sentence similarity task, Manhattan similarity metric can help to achieve faster convergence and higher accuracy than any other similarity metric. Our results also suggest that the Siamese architectures which are effective in English NLP tasks may not necessarily work well in Chinese NLP tasks.

In the future, we will try to build and conduct experiments on Siamese Transformer (Vaswani et al., 2017) architecture. In addition, we will use BERT (Devlin et al., 2018) to obtain word embeddings as the inputs of the Siamese architecture to test performances.

References

J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. 1993. Signature verification using a siamese time delay neural network. *Advances in Neural Information Processing Systems*, pages 737–744.

S. Chopra, R. Hadsell, and Y. LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 539–546.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 1993. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

F. Dai and C. Zheng Cai. 2017. Glyph-aware embedding of chinese characters. arXiv:1709.00028, 2017.

J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 1735–1742.

B. Hu, Z. Lu, H. Li, and Q. Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems*, pages 2042–2050.

G. Koch, R. Zemel, and R. Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, pages 148–157.

T. Mikolov, K. Chen I. Sutskever, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, pages 3111–3119.

J. Mueller and A. Thyagarajan. 2016. Dimensionality reduction by learning an invariant mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*.

P. Neculoiu, M. Versteegh, and M. Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157.

J. Pennington, R. Socher, and C. D. 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.

E. L. Pontes, S. Huet, A. C. Linhares, and J. M. Torres-Moreno. 2018. Predicting the semantic textual similarity with siamese cnn and lstm. arXiv:1810.10641.

T. Ranasinghe, C. Orasan, and R. Mitkov. 2019. Semantic textual similarity with siamese neural networks. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1004–1011.

T. Su and H. Lee. 2017. Learning chinese word representations from glyphs of characters. arXiv:1708.04755.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 5998–6008.

Y. Yang, Y. Zhang, C. Tar, and J. Baldridge. 2019. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. arXiv:1908.11828.

Automatic Classification of Students on Twitter Using Simple Profile Information

Christopher Wun*
Hunter College High School
New York, NY, 10128
ckywun123@gmail.com

Lili-Michal Wilson*
Hunter College High School
New York, NY, 10128
lilimmwilson@gmail.com

Abstract

Obtaining social media demographic information using machine learning is important for efficient computational social science research. Automatic age classification has been accomplished with relative success and allows for the study of youth populations, but student classification—determining which users are currently attending an academic institution—has not been thoroughly studied. Previous work (He et al., 2016) proposes a model which utilizes 3 tweet-content features to classify users as students or non-students. This model achieves an accuracy of 84%, but is restrictive and time intensive because it requires accessing and processing many user tweets. In this study, we propose classification models which use 7 numerical features and 10 text-based features drawn from simple profile information. These profile-based features allow for faster, more accessible data collection and enable the classification of users without needing access to their tweets. Compared to previous models, our models identify students with greater accuracy; our best model obtains an accuracy of 88.1% and an F1 score of .704. This improved student identification tool has the potential to facilitate research on topics ranging from professional networking to the impact of education on Twitter behaviors.

1 Introduction

In recent years, social media has become the focus of more scientific research studies (Khang et al., 2012), and is being used as a valuable means of understanding human tendencies and social circles. Due to rapid developments in technology, social media has become a hub for scientific communication and research (Rowlands et al., 2011), a source of entertainment and news, and a place for networking and socialization (Whiting and Williams, 2013). Twitter is one of the most popular social media platforms, especially among young people; 44% of young adults aged 18-24 and 31% of people aged 25-29 use Twitter, with 25% using it daily and 14% using it several times per day.[1]

The ability to group Twitter users by demographic information is crucial in understanding the behaviors of certain subsets of the population. Twitter provides limited information on user demographics, necessitating the development of machine learning models to predict non-explicit user features. Several studies have demonstrated that it is possible to classify users by gender (Knowles et al., 2016), age (Morgan-Lopez et al., 2017; Simaki and Iosif Mporas, 2018; Smith and Gaur, 2018), whether or not they are an organization (Wood-Doughty et al., 2018), and more.

In this paper, we aim to classify Twitter users as students or non-students. While many age classification models have been proposed, understanding this subset of the age classification problem may provide new insights into behavior on Twitter. Grouping users into students and non-students has applications ranging from professional networking (He et al., 2016) to understanding these users' role in the spread of misinformation, and by extension, the role that education plays in their online behaviors (Chen et al., 2015). Previous studies of students' social media interactions have obtained student samples using relatively low-yield, low-accuracy, or low-coverage techniques such as surveying individuals, analyzing geographic proximity to educational institutions, or manually annotating users one by one (Chen et al., 2015; Miller and Melton, 2014; Moreno et al., 2016; Veletsianos and Kimmons, 2016; Hanson et al., 2013). Research into automatic demographic prediction of student users could facilitate such student-based studies.

*Authors contributed equally

[1] https://www.pewresearch.org/fact-tank/2019/04/10.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 30–36

The most prominent application of an automatic classification model to student demographic research is described in He et al. (2016). This study develops a machine learning model that uses user tweet content to identify students on Twitter and match them with professional mentors on LinkedIn. Despite the model's relatively high accuracy (84%), its use of user tweet content is restrictive because it is not always possible to obtain complete tweet data. Protected accounts cannot be classified using this model, and researchers without advanced Twitter API access may not be able to obtain sufficient tweet data.

Here, we propose new models for student classification on Twitter which achieve high accuracy using only simple profile information. Our use of profile-based features obviates the need for tweet content, which allows for the classification of protected accounts and makes our model more accessible to researchers without access to advanced Twitter APIs. In addition, user profile information can be requested from the Twitter API at a faster rate than a sample of user tweets. These novel student identification models thus show potential for more accurate, accessible, and efficient classifications in demographic studies of social media users.[2]

1.1 Project Motivation

The development of this student identification classifier is part of a larger investigation into the way that students interact with misinformation on Twitter. Though the platform has implemented fact-checking guidelines, it is still considered one of the worst social media outlets in terms of spreading false information.[3] This spread of misinformation has necessitated research into ways of detecting rumors in tweets and predicting which kinds of users are most susceptible to resharing it (Vosoughi et al., 2018; Khan and Idris, 2018; Bodaghi et al., 2019; Margolin et al., 2017). This work is especially important in the current pandemic, where false claims on Twitter have pushed people to put themselves or others at risk (Barua et al., 2020; Rosenberg et al., 2020).

We hypothesize that users who are currently receiving education may behave differently on the platform due to a recent emphasis placed on fact checking at academic institutions.[4] Understanding if students play a different role in the spread of false information is relevant when determining what kinds of education can be implemented to prevent the spread of misinformation and how certain users could be targeted with that education.

Sample Type	S	N-S	Total
General stream	11	53	64
Hunter College	127	743	870
"Who to Follow"	86	8	94
Total	224	804	1028

Table 1: Dataset Distribution, Sample Types are defined in §2.1-2.3. S denotes student, while N-S denotes non-student.

1.2 Ethical Considerations

With the creation of a student classification model comes potential for misuse. Being able to determine whether or not a user is a student, even if their tweets are not public, could enable people to target student users with spam or other harmful content. As previously noted, students are generally a younger population who may be more vulnerable to online predators or scammers who are using this tool.

2 Data

Creating the dataset used to train the student identification model required obtaining a set of Twitter users who could be labeled as student or non-student and then manually labeling each user. These users were obtained from a combination of a general sample of Twitter users, a sample of the followers of a university Twitter account, and Twitter's "Who to Follow" feature. Ultimately, 1,037 users were labeled, with 225 students and 812 non-students. The distribution of the sample is shown in Table 1, with "S" representing students and "N-S" representing non-students. Users were manually labeled based on their Twitter bios, tweet content, or other online profiles (e.g. LinkedIn, Instagram).

A user was labeled as a student if their accounts indicated that they were in high school, undergraduate school, or graduate school in the spring of 2020. General examples of labeled students included users that provided a future graduation year

[2] Annotated datasets and student classifiers are publicly available at https://github.com/christopherwun/twitter-student-classifier

[3] https://www.washingtonpost.com/technology/2020/04/07/twitter-almost-60-percent-false-claims-about-coronavirus-remain-online-without-warning-label/

[4] https://www.nytimes.com/2020/02/20/education/learning/news-literacy-2016-election.html

in their bio; listed a current school in an associated LinkedIn account; or discussed homework, test scores, and/or their school's policies in their tweets.

A user was labeled as a non-student if their online profile showed that they had graduated from a higher institution, listed a full-time job, or were organizations or spam bots. General examples included noting a past graduation date or degree in their Twitter bio or listing jobs or graduation dates on an associated LinkedIn profile.

Labeling was conducted by the listed authors, and inter-annotator agreement was measured on a representative subset of 435 users. The subset demonstrated substantial agreement as shown by a Cohen's Kappa of 0.73 (McHugh, 2012). In the case of a disagreement, the first ascribed label was used in the final dataset. To avoid annotator bias in assigning these labels, each annotator labeled half of the dataset then reviewed the other annotator's labels blind.

2.1 General Twitter Stream

In order to get a sample of the overall Twitter population, we collected tweets written in English from the Twitter stream for ten minutes. This general stream was used to obtain a variety of users across Twitter for labeling. However, the proportion of identifiable students within this set was low, necessitating other sampling to increase the number of student examples.

2.2 College Followers

In order to increase the number of students in the dataset, followers of @Hunter_College (the verified account of a prominent NYC college) were sampled directly and manually labeled. This sample was taken under the assumption that the pool of followers of this school account would contain an increased proportion of students.

2.3 "Who to Follow": Similar Users

Finally, to further increase the number of students in the set, Twitter's "Who to Follow" feature was applied to students that had been identified through the general stream and Hunter College followers. The "Who to Follow" feature identifies users similar to a given user. When applied to previously labeled students, it often identified potential students with similar roles in the social network. These potential students were then labeled based on the same criteria as the previous two samples.

2.4 Limitations

Approximately 50% of sampled users could not be identified with certainty as a student or a non-student, and such users were not labeled and not included in the dataset.

Another potential issue with the dataset is its relatively small size, due to the time-intensive manual labeling process. In addition, the methods used to upsample students led to an uneven distribution of student users across data sources (Table 1). It is important to keep these limitations in mind when considering the results of our model.

3 Methods

3.1 Feature Extraction

A combination of metadata-based features and custom text-based features was used to train the models (Table 2). For users without descriptions (approximately 7.4% of the dataset), zeros were recorded for all description-based features. This includes both explicitly student-leaning features like "Student?" and "Year?" as well as explicitly non-student-leaning features like "Alum?" and "Occupation?". Features were scaled to similar ranges using scikit-learn's StandardScaler (Pedregosa et al., 2011) in order to improve model performance.

Unlike other student-identifying models, only profile information was incorporated into this model (i.e., no tweet data or other user data was considered). User features were extracted through the Twitter API in batches of 100 user IDs. The Twitter API limits requests to 900 per 15 minutes, meaning that our method allows 90,000 users to be extracted per 15 minutes.[5] A tweet-content-based approach such as the one used in He et al. (2016) must acquire user features by accessing their tweets, and requests for tweets from different users' timelines cannot be batched. Therefore, a tweet-based method allows the features of only 900 individual users to be extracted per 15 minutes.

3.2 Feature Selection

Twenty original features were extracted from each user, and three were removed due to low importance to the machine learning models (See Table 2). Importance was assessed via logistic regression importance rankings, decision tree rankings, random forest rankings, and LASSO rankings. This feature removal was verified via an ablation study, which

[5] https://developer.twitter.com/en/docs/twitter-api/rate-limits

Feature Name	Value Type	Feature Description
Student?	Binary	Has “student”, “estudiante”, or “studying” in description. Not “students”.
Friends	Continuous Numerical	Number of users followed by the account.
Occupation?	Binary	Has an occupation in description in occupation dataset.[6] Not “aspiring” or “future”.
Emojis	Continuous Numerical	Number of emojis in user description squared.
Liked Posts	Continuous Numerical	Number of posts the account has ’liked’.
Parent?	Binary	Has “mom”, “mama”, “mother”, “dad”, “papa”, or “father” in description.
Consecutive Upper	Continuous Numerical	Number of cons. uppercase letters in screen name.
Name Emojis	Continuous Numerical	Number of emojis in screen name.
Name Title	Binary	Has “mr.”, “ms.”, “mrs.”, “ph.d”, “ph. d”, “phd”, “m.d”, “m. d”, “doctor”, or “dr.” in screen name.
Link?	Binary	Has an associated link.
Tweet Rate	Continuous Numerical	Tweets posted per year by this account.
Tweet Count	Continuous Numerical	Total number of tweets (including retweets) posted by this account.
Year?	Binary	Has “ ‘2_ ”, “ 2_’ ”, “class of 202x”, “freshman”, or “sophomore” in description.
Followers	Continuous Numerical	Number of users following the account.
Alum?	Binary	Has “alum” in description.
Views My Own?	Binary	Has “(views)/(opinions) (mine)/(my own)” or “rts not endorsements” in description.
Verified?	Binary	Has been verified.
Created At*	Continuous Numerical	Timestamp for creation time of account.
Account Age*	Continuous Numerical	Time (in years) since creation time of account.
Last Tweet Time*	Continuous Numerical	Time (in years) since the account’s last tweet.

Table 2: Profile-Based Features (ordered by importance)
*Indicates feature was removed from the final model

showed that all remaining features had a positive importance averaged across all six models (Fig. 1).

3.3 Model Selection

We looked into six different machine learning models implemented in scikit-learn (Pedregosa et al., 2011) to create our student identification classifier: 1) Logistic Regression, 2) Random Forest, 3) SVM, 4) K-Nearest Neighbors, 5) AdaBoost (a Random Forest model retrained multiple times, each time increasing the weight of previously misclassified users), and 6) a Stacked Classifier (a Logistic Regression model using the outputs of models 1-4 as inputs). We performed a grid search of model hyperparameters with 10-fold cross-validation to optimize each model for the highest F1 score.

After noticing that models placed a heavy emphasis on the “Student?” feature (Fig. 1), a simple

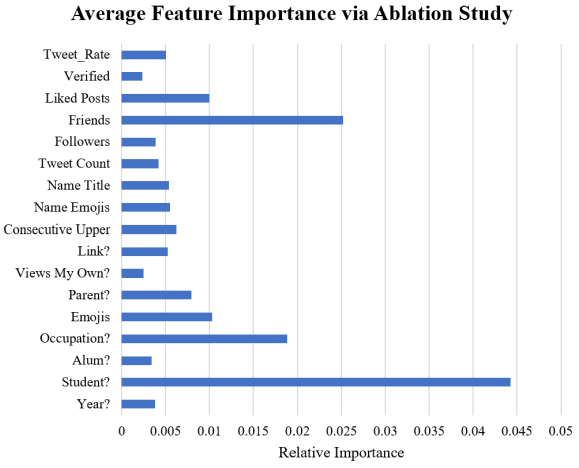


Figure 1: Relative Importance of User Features (excluding removed features)

[6]https://gist.github.com/wsc/1083459

"if-statement classifier" was created to determine whether or not the machine learning models were adding anything to the classification. It labeled a user as a student if the "Student?" feature was equal to 1. We also created a tweet-content-based SVM model to use as a baseline (described in §4.1).

3.4 Model Tuning

After selecting the top three model types based on F1 score and AUROC (defined in §4), we further improved our results by adding regions within our prediction probabilities, which we call "gray areas," where our models would classify a user as "uncertain." These regions were identified by testing 39 candidates with 10-fold cross-validation. Two gray area candidates were selected based on accuracy and F1 score for each of the three model types.

4 Results

Table 3 reports the results from running the six different classifiers and two baseline models without implementing a gray area. The test set reflected the distribution of the overall dataset, with a ratio of students to non-students of around 1:4. We report the accuracy (percent of correctly predicted students or non-students over the total), F1 score (the harmonic mean of precision and recall), and area under the Receiver Operator Characteristic curve (AUROC; an ROC curve plots the true positive rate against false positive rate at different decision boundaries).

Before considering gray area, the Stacked Classifier outperformed all other classifiers in accuracy, F1, and AUROC. Every model outperformed the "if-statement classifier" in both accuracy and F1 score, indicating that the additional complexity and extra features considered in the machine learning models contributed to their success.

Table 4 reports the results from running the top three classifiers (Stacked Classifier, Logistic Regression, and Random Forest) using two gray area boundaries that were identified through 10-fold cross validation as described in §3.3. AUROC is not reported here as it is the same in Table 3. Instead, we report coverage, which is the percentage of test examples that the gray area model labeled. The thresholds for the gray area are reported in the table as (lower bound, upper bound). Of these gray area models, the Stacked Classifier with thresholds at (0.25, 0.55) outperformed the rest, with an accuracy of 90.3% and an F1 of .761. The model also retained a relatively high coverage, labeling nearly 90% of the test examples.

	Accuracy	F1	AUROC
Logistic Regression	87.4	.678	.896
Random Forest	86.7	.677	.910
SVM	86.8	.643	.898
KNN	81.6	.601	.750
AdaBoost	87.1	.683	.785
Stacked Classifier	**88.1**	**.704**	**.917**
If-Statement	80.0	.340	-
Tweet-based SVM	77.1	.222	.646

Table 3: Model Comparison

It is important to note that, while the gray area models outperform the normal models in both accuracy and F1, much of that can be attributed to the fact that certain false negative examples are being ignored. Depending on the intended use, a higher accuracy version of the Stacked Classifier may be used sacrificing coverage, or vice versa.

	Accuracy	F1	Coverage
Logistic Regression (0.3, 0.4)	89.6	.760	89.7
Logistic Regression (0.35, 0.45)	89.8	.748	91.3
Random Forest (0.3, 0.5)	89.4	.749	88.4
Random Forest (0.35, 0.45)	87.4	.713	94.5
Stacked Classifier (0.25, 0.55)	**90.3**	**.761**	**89.7**
Stacked Classifier (0.35, 0.55)	89.3	.735	93.2

Table 4: Gray Area Model Comparison

4.1 Comparison to Previous Models

In order to better understand how our model compares to previous student identification attempts, we recreated the tweet content classification model described in He et al. (2016). They used the relative frequency of three expressions—"HAHA"/"LOL", emojis, and hashtags—among the 200 most recent user tweets as features in their LIBSVM model. We trained and tested this model on our labeled dataset, then optimized the model's hyperparame-

ters as described in §3.3. The tuned model achieved an accuracy of 77.1%, an F1 score of 0.222 and an AUC of 0.646 (Table 3). The accuracy of this model was lower than the reported 84% in He et al. (2016). The F1 and AUC scores cannot be compared since they were not reported in the original paper.

A likely explanation for the inconsistency in accuracy lies in dataset construction. He et al. (2016) partially automated their data annotation by using regular expressions from tweets to identify student candidates before manually labeling these users. This technique may have increased the proportion of high-tweeting users in their dataset and made their tweet-based approach to classification more effective. In contrast, our manual annotations did not rely on vast amounts of tweet content to assign a label. Thus, our model is more reliable when analyzing users with lower tweet counts.

Our profile-based student classifier outperformed the tweet-content classifier in accuracy, F1 score, and AUROC when applied to our dataset. Therefore, this model improves upon existing models and can be used as a tool in future student demographic research.

4.2 Error Analysis

Organizations that are tailored towards students and university accounts commonly appear among the misclassified users. In order to mitigate this misclassification, our student classifier could be used in conjunction with an individual versus organization classifier similar to the one presented in Wood-Doughty et al. (2018). Organizations could be filtered out before applying the student classifier, and the model's false positive rate would likely decrease.

Among the 24 users without descriptions (7.7% of the test set), only 2 of them were misclassified: one student, and one non-student. The model's accuracy is thus comparable between users without descriptions (accuracy = 91.7%) and the general test set (accuracy = 88.1%). As the sample size is small, we cannot conclude that users without descriptions are labeled more accurately than the general set; however, this result does indicate that users without descriptions can still be accurately classified by our model.

5 Conclusions

In this paper, we introduce a metadata-based machine learning model to accurately predict student Twitter users. We also introduce a gray-area model that achieves 90.3% accuracy without leaving many users unlabeled. Our models improve upon past research by providing more accurate, more efficient, and faster classifications due to their use of only simple profile information.

Currently, we are working to apply this student classifier in a preliminary study of student interactions with COVID-19 related misinformation on Twitter.

Acknowledgments

We thank Zach Wood-Doughty for his guidance on feature selection and model optimization, and the AACL-IJCNLP SRW reviewers and mentors for their comments and suggestions. We would also like to thank Gilana Reiss and Philip Frankel for their support during the project conception.

References

Zapan Barua, Sajib Barua, Salma Aktar, Najma Kabir, and Mingze Li. 2020. Effects of misinformation on covid-19 individual responses and recommendations for resilience of disastrous consequences of misinformation. *Progress in Disaster Science*.

A. Bodaghi, S. Goliaei, and M. Salehi. 2019. The number of followings as an influential factor in rumor spreading. *Applied Mathematics and Computation*, 357:167–184.

Xinran Chen, Sei-Ching Joanna Si, Yin-Leng Theng, and Chei Sian Lee. 2015. Why students share misinformation on social media: Motivation, gender, and study-level differences. *The Journal of Academic Librarianship*, 41.

Carl L Hanson, Scott H Burton, Christophe Giraud-Carrier, Josh H West, Michael D Barnes, and Bret Hansen. 2013. Tweaking and tweeting: Exploring twitter for nonmedical use of a psychostimulant drug (adderall) among college students. *Journal of Medical Internet Research*, 15(4):e62.

Ling He, Lee Murphy, and Jiebo Luo. 2016. Using social media to promote STEM education: Matching college students with role models. In *Machine Learning and Knowledge Discovery in Databases*, pages 79–95. Springer International Publishing.

M. Laeeq Khan and Ika Karlina Idris. 2018. Recognise misinformation and verify before sharing: a reasoned action and information literacy perspective. *Behaviour Information Technology*, 38:1194–1212.

Hyoungkoo Khang, Eyun-Jung Ki, and Lan Ye. 2012. Social media research in advertising, communication, marketing, and public relations, 1997–2010. *Journalism Mass Communication Quarterly*, 89:279–298.

Rebecca Knowles, Josh Carroll, and Mark Dredze. 2016. Demographer: Extremely simple name demographics. In *Proceedings of the First Workshop on NLP and Computational Social Science*. Association for Computational Linguistics.

Drew B. Margolin, Aniko Hannak, and Ingmar Weber. 2017. Political fact-checking on twitter: When do corrections have an effect? *Political Communication*, 35:196–219.

Marry L. McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia Medica*, pages 276–282.

Robert Miller and James Melton. 2014. College students and risk-taking behaviour on twitter versus facebook. *Behaviour Information Technology*, 34:678–684.

Megan A. Moreno, Alina Arseniev-Koehler, Dana Litt, and Dimitri Christakis. 2016. Evaluating college students' displayed alcohol references on facebook and twitter. *Journal of Adolescent Health*, 58(5):527–532.

Antonio A. Morgan-Lopez, Annice E. Kim, Robert F. Chew, and Paul Ruddle. 2017. Predicting age groups of twitter users based on language and metadata features. *PLoS One*, 12(8).

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.

Hans Rosenberg, Shahbaz Syed, and Salim Rezaie. 2020. The twitter pandemic: The critical role of twitter in the dissemination of medical information and misinformation during the covid-19 pandemic. *Canadian Journal of Emergency Medicine*, 22:418–421.

Ian Rowlands, David Nicholas, Bill Russell, Nicholas Canty, and Anthony Watkinson. 2011. Social media use in the research workflow. *Learned Publishing*, 24:183–195.

Vasiliki Simaki and Vasileios Megalooikonomou Iosif Mporas. 2018. Age identification of twitter users: Classification methods and sociolinguistic analysis. *Computational Linguistics and Intelligent Text Processing*, pages 385–395.

Alan Smith and Manas Gaur. 2018. What's my age?: Predicting twitter user's age using influential friend network and dbpedia. arXiv:1804.03362v1.

George Veletsianos and Royce Kimmons. 2016. Scholars in an increasingly open and digital world: How do education professors and students use twitter? *The Internet and Higher Education*, 30:1–10.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359:1146–1151.

Anita Whiting and David Williams. 2013. Why people use social media: a uses and gratifications approach. *Qualitative Market Research*, 16:362–369.

Zach Wood-Doughty, Praateek Mahajan, and Mark Dredze. 2018. Johns hopkins or johnny-hopkins: Classifying individuals versus organizations on twitter. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*. Association for Computational Linguistics.

Towards Code-switched Classification Exploiting Constituent Language Resources

Kartikey Pant * and **Tanvi Dadu** *
International Institute of Information Technology, Hyderabad, India
Netaji Subhas Institute of Technology, New Delhi, India
kartikey.pant@research.iiit.ac.in
tanvid.co.16@nsit.net.in

Abstract

Code-switching is a commonly observed communicative phenomenon denoting a shift from one language to another within the same speech exchange. The analysis of code-switched data often becomes an assiduous task, owing to the limited availability of data. We propose converting code-switched data into its constituent high resource languages for exploiting both monolingual and cross-lingual settings in this work. This conversion allows us to utilize the higher resource availability for its constituent languages for multiple downstream tasks.

We perform experiments for two downstream tasks, sarcasm detection and hate speech detection, in the English-Hindi code-switched setting. These experiments show an increase in 22% and 42.5% in *F1-score* for sarcasm detection and hate speech detection, respectively, compared to the state-of-the-art.

1 Introduction

Code-switching is the juxtaposition of speech belonging to two different grammatical systems or subsystems within the same speech exchange. In other words, it denotes a shift from one language to another within the same utterance but not limited to beyond mere insertion of borrowed words, fillers, and phrases. It often involves morphological and grammatical mixing(Sitaram et al., 2019). Code-switching of speech is a common phenomenon in the multilingual communities (Hickey, 2012), especially among peers who have similar fluency in multiple languages. Code-switching among languages like Spanish-English, Hindi-English, Tamil-English is often seen to be used by bilingual speakers of these languages.

Code-switching is likely to be found in informal settings, including speech and social media, and semi-formal such as navigation instructions. This phenomenon leads us to acknowledge code-switching as a legitimate communication form deserving careful linguistic analysis and processing. Processing code-switched communication would enhance user experience in various industrial settings, including advertising, healthcare, education, and entertainment (Sitaram et al., 2019).

Access to code-switched data is challenging and limited. This phenomenon makes the analysis and information extraction from code-switched languages a less explored and challenging task. Most code-switching studies focus on pairs of a high resource and a lower resource language, often making these studies data starved. Also, code-switched language is mostly used for informal and semi-formal settings, making it less archived, resulting in its limited availability in terms of context and volume.

Traditionally, most works in academia have limited themselves to language-independent methodologies like SVM, FastText, and CNN (Sitaram et al., 2019) for modeling code-switched languages. This can be attributed to the shortage of data in code-switched settings, limiting the exploitation of predictive performance achieved via transfer learning using pretrained contextualized word embeddings.

We propose a generic methodology for modeling Hindi-English code-switched data revolving around its translation and transliteration. We propose two approaches that exploit a) Transliteration to convert this low-resource task into a high-resource task using cross-lingual learning, and b) Translation to utilize monolingual resources of each of the constituent languages. The importance of this methodology is that we enable the exploitation of pretrained transformer-based embeddings that are available in the monolingual or cross-lingual setting but not code-mixed setting

* Both authors contributed equally to the work.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 37–43

due to the scarcity of data for pretraining. We then benchmark both the approaches for the tasks of hate speech detection and sarcasm detection using the current state-of-the-art and language-independent supervision learning as the baseline. We further provide insights into the proposed approaches and compare them analytically.

2 Related Works

The previous works on sarcasm and humor detection contain a myriad of methods employed over the years on multiple datasets, including statistical and N-gram analysis on spoken conversations from an American sitcom (Purandare and Litman, 2006), Word2Vec combined with KNN Human Centric Features on the *Pun of the Day* and *16000 One-Liner datasets* (Yang et al., 2015), and Convolutional Neural Networks on datasets with distinct joke types in both English and Chinese (Chen and Soo, 2018). Recently, transformer-based architectures have been used to detect humor on multiple datasets collected from Reddit, Kaggle, and Pun of the Day Dataset (Weller and Seppi, 2019).

The prior works exploring hate speech detection employ several machine learning-based methods like linguistic features with character n-grams (Waseem and Hovy, 2016), SVM with a set of features that includes n-grams, skip-grams, and clustering-based word representations (Malmasi and Zampieri, 2017). Neural networks like LSTM (Badjatiya et al., 2017), FastText (Badjatiya et al., 2017) and CNN (Gambäck and Sikdar, 2017) have also been used for hate speech detection. Recently, transformer-based architectures using BERT, ALBERT, and RoBERTa have been used to detect hate speech (Mozafari et al., 2019; Wiedemann et al., 2020).

There has been considerable work in code-switching, as is well documented by Sitaram et al. (2019). The use of code-switched data has been explored for the tasks of Part-of-Speech tagging (Vyas et al., 2014), Named Entity Recognition (Aguilar et al., 2018), Dependency Parsing (Partanen et al., 2018), and Text Classification (Swami et al., 2018; Bohra et al., 2018).

Swami et al. (2018) introduced an English-Hindi code-switching dataset for sarcasm detection in social media text. Their dataset contains tweets annotated as sarcastic and non-sarcastic in the form of their constituent language-identified tokens. They further provide a feature-based model as a baseline. We use their dataset for our sarcasm detection experiments.

Bohra et al. (2018) introduced a dataset of English-Hindi social media text, demonstrating code-switching for the task of hate speech detection. They also conduct benchmarking experiments using a feature-based model. Similar to Swami et al. (2018), their benchmarks only contain accuracies for their baseline systems and do not take F1-scores into account. We use their dataset for our hate speech detection experiments.

There have been attempts to exploit monolingual datasets for enhancing downstream tasks in a code-switched setting. Solorio and Liu (2008) explored the use of monolingual taggers for Spanish and English for Part-of-Speech Tagging in an English-Spanish code-switched dataset. They further show that their best results were obtained by using the output of monolingual POS taggers as input features. Gupta et al. (2018) explore using monolingual and bilingual resources for the task of Question Answering in an English-Hindi code-switched dataset.

For text classification involving code-switched text, traditional methodologies like SVM, FastText, CNN has been used (Sitaram et al., 2019). However, most works are unable to exploit the recent advances in pretrained models, including RoBERTa and XLM-RoBERTa (Liu et al., 2019; Ruder et al., 2019) due to data scarcity and lack of normalization in the code-switched setting. To overcome this, Srivastava and Vardhan (2020) attempt to use multilingual BERT for code-switched sentiment analysis of social media text using a transliteration based methodology. However, they failed to beat the traditional baselines, including fasttext.

3 Approach

This section outlines our approach for analyzing code-switched language by converting it into its respective high resource languages and fine-tuning monolingual and cross-lingual contextual word embeddings. We highlight the data preparation followed by the contextual embedding used for each of the variants. Figure 1 illustrates our proposed approach through an example code-switched sentence from one of the datasets.

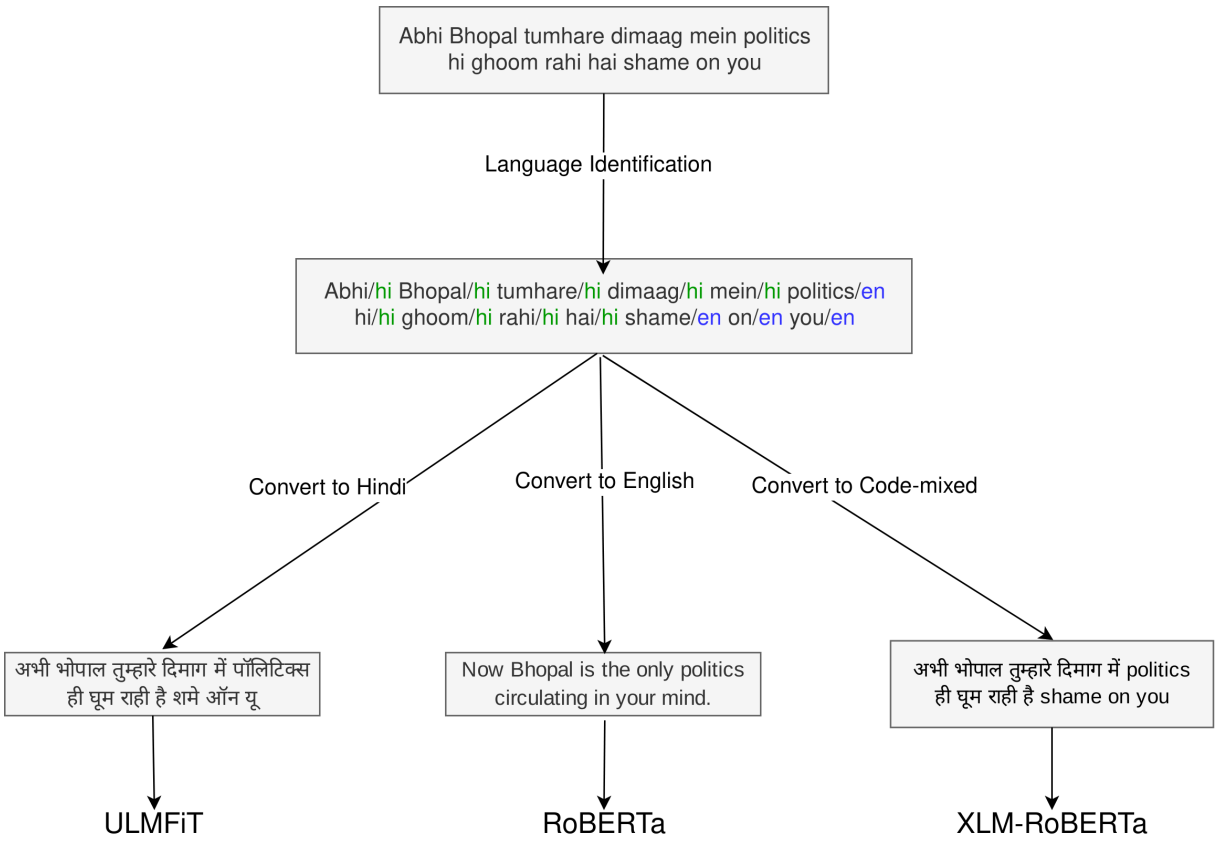


Figure 1: An example sentence demonstrating the proposed approach.

3.1 High Resource Language A - Hindi

3.1.1 Data Preparation

This section outlines our approach for analyzing code-switched language by converting it into its respective high resource languages and fine-tuning monolingual and cross-lingual contextual word embeddings. We highlight the data preparation followed by the contextual embedding used for each of the variants.

This section presents the steps performed for converting the code-switched data into one of the high resource languages, Hindi:

1. **Language Identification**: Each token was identified as either containing Hindi, English, or neither of them.

2. **Transliteration into Devanagari**: Each language-marked token, containing Hindi and English words, was transliterated from the Latin script to the Devanagari script. We use Microsoft Translate Text [1] service's Transliteration API for the transliteration process.

[1] https://www.microsoft.com/en-us/translator/business/translator-api/

Language Identification for the Hate Speech Detection task was done using a supervised FastText model trained on the Sarcasm Detection corpus (Swami et al., 2018) containing tokens with their respective languages- English, Hindi, and Other (includes Hashtags, Mentions, and Punctuation marks).

Dataset	Percentage of Hindi Tokens
Sarcasm Detection	88.259%
Hate Speech	86.437%

Table 1: Percentage of Hindi Tokens in comparison to Language-marked Tokens.

Table 1 shows that the majority of the words present in the dataset are Hindi, allowing us to assume that code-switched language contains Hindi and English words borrowed into Hindi. This phenomenon allows us to take advantage of converting all words from Latin to Devanagari via transliteration.

3.1.2 Classification Model

For the classification of the text translated to Hindi, we use Universal Language Model Fine-tuning for Text Classification, ULMFiT (Howard and Ruder,

2018). We use a pretrained language model trained on $172K$ Wikipedia articles in Hindi, attaining a perplexity of 34.06 [2]. We fine-tune our language model using discriminative fine-tuning, using different learning rates for different layers of the language model and slanted triangular learning rates. We then fine-tune the final model with the text translated in Hindi, after augmenting the trained language model with two additional linear blocks for classification. These blocks use batch normalization and dropout, with ReLU activations for the intermediate layer and a softmax activation for the last layer.

3.2 High Resource Language B - English

3.2.1 Data Preparation

This section outlines the conversion process of the code-switched data into the other high resource language, English:

1. **Language Identification**: Similar to the previous subsection, each token was identified to be either containing Hindi, English, or neither of them.

2. **Transliteration into Devanagari**: We used the same model as in the previous subsection for transliterating each language-marked token from the Latin script to the Devanagari script.

3. **Translation into English**: Each full sentence from the transliterated text was translated into English using the Translate API of the Microsoft Translate Text service.

3.2.2 Classification Model

For classification of the generated English text, we use a strong model for monolingual English classification, $RoBERTa_{Large}$. RoBERTa (Liu et al., 2019) is a contextualized word representation model, pre-trained using a bidirectional Transformer-based encoder. It is trained using significantly larger data with carefully tuned hyperparameters and performs competitively in benchmarking texts like GLUE (Wang et al., 2018) for text classification and SQuAD (Rajpurkar et al., 2016) for question answering. Since the model is pretrained on a large generic dataset, it can be fine-tuned for a specific task in a simpler yet efficient manner without making significant changes in its architecture. We fine-tune its large variant, $RoBERTa_{Large}$, for both the tasks.

[2] https://github.com/goru001/nlp-for-hindi

3.3 Cross-lingual Classification

3.3.1 Data Preparation

Each language-marked token needs to have its correct script to utilize cross-lingual word representations. We, therefore, employ the following steps to prepare the dataset for cross-lingual classification:

1. **Language Identification**: Similar to the previous subsections, each token was identified to be either containing Hindi, English, or neither of them.

2. **Transliteration of Hindi words into Devanagari**: We used the same model as in the previous subsections for transliterating each token marked to be Hindi from the Latin script to the Devanagari script.

3.3.2 Classification Model

For cross-lingual classification, we use XLM-RoBERTa, a state-of-the-art contextual word representation released by Facebook (Ruder et al., 2019). XLM-R is a cross-lingual unsupervised contextual word representation pretrained on a 100-language sized dataset. We exploit its advantage of massive cross-lingual transfer learning by fine-tuning it for both the tasks. Due to its high scalability and ease of use in fine-tuning, the model enables us to achieve competitive predictive performance while using minimum training resources. We use the large variant, XLM-R, for both the tasks.

4 Experimental Results

4.1 Downstream Tasks

This section introduces the datasets used for the downstream tasks of Sarcasm Detection and Hate Speech Detection.

For the task of sarcasm detection, we use English-Hindi code-switched dataset released by Swami et al. (2018). The dataset contains $5,250$ tweets sampled from Twitter with sarcastic or non-sarcastic tags. It also provides language tags (English/Hindi/Others) for all the tokens present in tweets. The dataset has 0.790 Cohen's Kappa score for measuring inter-annotator agreement, implying a high quality of the annotation schema.

For the task of hate speech detection, we use Bohra et al. (2018)'s dataset containing tweets showing English-Hindi code-switching. From their

released dataset, we were able to get $4,575$ tweets along with their binary label denoting whether the tweet contains hate speech or not. We preprocess the dataset and remove hyperlinks embedded in the tweets after language identification of each token. The authors measure the inter-annotator agreement through Kohen's Kappa score, which is reasonably high, with a value of 0.982.

4.2 Baselines

We use the following neural architectures as baselines for our experiments:

1. FastText (Joulin et al., 2016): FastText is a library released by Facebook which uses bag of words and bag of n-grams as features for text classification. It relies on capturing partial information about the local word order efficiently.

2. Convolutional Neural Networks (CNN) (Kim, 2014): Convolutional neural networks are multistage trainable neural network architectures developed for classification tasks employing convolution and pooling for extracting features from the text.

4.3 Experimental Settings

In this subsection, we outline the experimental settings used for each of the models used in the experiment. We evaluated our model on a held-out test dataset for all experiments, consisting of 10% of the total dataset. For validation purposes, we further split our training dataset using a $90:10$ train-validation split. We evaluate all the models on the following metrics: *F1*, *Precision*, *Recall*, *Accuracy*.

We use FastText's recently open-sourced automatic hyperparameter optimization functionality and run 100 trials of optimization. We use a two-dense-layered architecture for Convolutional Neural Network with 3 and 4 convolution layers for sarcasm and hate speech detection, respectively. We further used a sequence length of 100 and an embedding size of 300 with a dropout rate of 0.1 and 0.2.

For ULMFiT, we use a pretrained language model trained on $172K$ Wikipedia articles in Hindi. We use the SentencePiece tokenizer for tokenizing the texts. For language modeling, we used a batch size of 16 and BPTT of 70. We use AWD-LSTM architecture with a dropout rate of 0.5 as a classifier. For $RoBERTa_{Large}$ and XLM-R, we fine-tune with a learning rate of $1 * 10^{-5}$ for 3 epochs, each with a maximum sequence length of 50.

4.4 Results

Table 2 and Table 3 show the experimental results obtained for the sarcasm detection dataset and hate speech dataset, respectively. We observe that our proposed approach of converting code-switched data into respective high resource language outperforms previously used approaches like FastText and CNN significantly. For both sarcasm detection and hate speech detection, preparing code-switched data for cross-lingual classification and using it to fine-tune XLM-R gives the best results. This is reflected in the model obtaining an *F1-score* of 0.850 for sarcasm detection and 0.724 for hate speech detection. For both tasks, respectively, it is 22% and 42.5% higher than *CNN*, the best performing baseline.

Model	Accuracy	Precision	Recall	F1
FastText	76.22%	0.245	0.951	0.390
CNN	95.71%	**0.806**	0.610	0.694
ULMFiT	94.85%	0.800	0.733	0.765
$RoBERTa_{Large}$	95.99%	0.791	0.883	0.835
XLM-R	96.37%	**0.806**	**0.900**	**0.850**

Table 2: Experimental results for the sarcasm detection task.

Model	Accuracy	Precision	Recall	F1
FastText	68.78%	0.609	0.417	0.495
CNN	58.08%	0.411	0.664	0.508
ULMFiT	68.78%	0.545	0.506	0.525
$RoBERTa_{Large}$	71.62%	0.707	0.716	0.711
XLM-R	**71.83%**	**0.730**	**0.718**	**0.724**

Table 3: Experimental results for the hate speech detection task.

Further we observe that our proposed approach performs significantly well for all the metrics taken under consideration with stark increase in *F1-score*. In sarcasm detection, it outperforms *F1-score*, *Precision* where as in hate speech detection, it outperforms *F1-score*, *Precision*, *Recall* and *Accuracy* by a significant margin. In sarcasm detection, a slight increase in *Accuracy* can be attributed to the imbalance distribution of classes, which is reinforced by a significant increase in *F1-score*.

The experimental results obtained for both sarcasm and hate speech detection show our approach's effectiveness in leveraging the pretrained contextualized word embeddings for the code-switched settings. It also successfully tackles data

shortage when dealing with code-switched data in pretrained settings by converting code-switched language into its constituent languages.

5 Conclusion

In this work, we show that converting code-switched data into its constituent high resource language enables us to exploit the performance of the pretrained models for those languages on downstream tasks. We perform experiments on the English-Hindi code-switching pair and benchmark our approach for sarcasm detection and hate speech detection. The experiments show an improvement of up to 42.5% in *F1-score* compared to strong baselines like CNN. Our findings pave the way for further research to utilize monolingual resources for code-switched data for multiple downstream tasks and extend this methodology for other code-switched language pairs.

Acknowledgments

We would like to thank Sai Krishna Rallabandi for reviewing the drafts and the mentoring. We would also like to thank the anonymous reviewers for providing critical suggestions.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

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, page 759–760, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *NAACL-HLT (2)*. Association for Computational Linguistics.

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.

Vishal Gupta, Manoj Chinnakotla, and Manish Shrivastava. 2018. Transliteration better than translation? answering code-mixed questions over a knowledge base. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 39–50, Melbourne, Australia. Association for Computational Linguistics.

Raymond Hickey. 2012. *The Handbook of Language Contact*, volume 1. 'John Wiley & Sons'.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. Cite arxiv:1607.01759.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

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.

Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A bert-based transfer learning approach for hate speech detection in online social media. In *COMPLEX NETWORKS*.

Niko Partanen, Kyungtae Lim, Michael Rießler, and Thierry Poibeau. 2018. Dependency parsing of code-switching data with cross-lingual feature representations. In *Proceedings of the Fourth International Workshop on Computational Linguistics of Uralic Languages*, pages 1–17, Helsinki, Finland. Association for Computational Linguistics.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for F*R*I*E*N*D*S*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215, Sydney, Australia. Association for Computational Linguistics.

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, Austin, Texas. Association for Computational Linguistics.

Sebastian Ruder, Anders Søgaard, and Ivan Vulić. 2019. Unsupervised cross-lingual representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 31–38, Florence, Italy. Association for Computational Linguistics.

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W. Black. 2019. A survey of code-switched speech and language processing. *CoRR*, abs/1904.00784.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 1051–1060, USA. Association for Computational Linguistics.

Aditya Srivastava and V. Harsha Vardhan. 2020. Hcms at semeval-2020 task 9: A neural approach to sentiment analysis for code-mixed texts.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A corpus of english-hindi code-mixed tweets for sarcasm detection. *ArXiv*, abs/1805.11869.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979, Doha, Qatar. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. 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, San Diego, California. Association for Computational Linguistics.

Orion Weller and Kevin D. Seppi. 2019. Humor detection: A transformer gets the last laugh. In *EMNLP/IJCNLP*.

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. UHH-LT & LT2 at semeval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection. *CoRR*, abs/2004.11493.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard H. Hovy. 2015. Humor recognition and humor anchor extraction. In *EMNLP*. The Association for Computational Linguistics.

Hindi History Note Generation with Unsupervised Extractive Summarization

Aayush Shah*, **Dhineshkumar Ramasubbu***, **Dhruv Mathew***, **Meet Chetan Gadoya***
University of Southern California
Los Angeles, California
{aayushsh, dramasub, dkmathew, gadoya}@usc.edu

Abstract

In this work, the task of extractive single document summarization applied to an education setting to generate summaries of chapters from grade 10 Hindi history textbooks is undertaken. Unsupervised approaches to extract summaries are employed and evaluated. TextRank, LexRank, Luhn and KLSum are used to extract summaries. When evaluated intrinsically, Luhn and TextRank summaries have the highest ROUGE scores. When evaluated extrinsically, the effective measure of a summary in answering exam questions, TextRank summaries performs the best.

1 Introduction

Our task is to apply text summarization to generate notes for school students where the medium of instruction is Hindi. The motivation for this work is that students studying under the Indian Central Board of Secondary Education (CBSE) have a lack of additional resources given their medium of instruction. Online resources are limited, with most reference guide material being published in English. Given the vast quantities of information that students are made to memorize, we believe that our tool will help provide students with an outline, a text summary, that could serve as both a big picture introduction and a pre-exam study guide. We focus on this task as each year over 18 million students give the grade ten exam. As Hindi is a low resource language, we believe that such a tool could help students learn better.

From prior research (Verma et al., 2019) on comparative text summarization in English and Hindi, we see that summarization results vary drastically for different languages and subject matters. While extensive research has been done for summarization techniques in English (Luhn, 1958; Edmundson, 1969; Carbonell and Goldstein, 1998; Pal and Saha, 2014; Erkan and Radev, 2004), directly applying said methods to Hindi text performs poorly (Verma et al., 2019). Frequency, graph and feature based approaches have been investigated previously to extract summaries from Hindi text and have shown to perform well on news documents (Vijay et al., 2017). Rule based methods (Gupta and Garg, 2016), and improvements to graph based methods incorporating semantic information from the text (Kumar et al., 2015) perform well for Hindi documents from various domains.

We wish to address the task of extractive text summarization in Hindi as it applies to learning history in an education setting for school students using unsupervised algorithms. The main reason behind choosing unsupervised methods for this task is that these algorithms do not require a dedicated training set annotated by individuals with subject specific knowledge. Secondly, employing a supervised approach for a particular domain constrains the portability of the trained model to be applied on different domains. Furthermore, the efficiency or goodness of the generated summaries for a particular task rely on accurate and reliable human annotated summaries used for training.

To the best of our knowledge, there exists no work that addresses Hindi text summarization in the academic domain as a note generating tool for students. This made it difficult to compare our approaches with existing work that deals with different domains of text data. In this work, we investigate unsupervised graph, term frequency and probability based single document summarization methods. Our work will build on previous linguistic analyses (for instance, no direct way to identify proper nouns) in Hindi (Paul et al., 2013) to deal with the nuances of summarizing history written in Hindi (Garg et al., 2012). Our code is publicly available on GitHub[1].

*These authors contributed equally to this work.

[1]Code: https://github.com/dhineshkumar-r/Unsupervised-Extractive-Summarization-Hindi-Note-Generation

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 44–49

2 Materials

We used the grade 10 Hindi history textbook (NCERT, 2018-2019) prescribed by the CBSE and published by the National Council of Educational Research and Training (NCERT) as the dataset. The Textbook is available in PDF format and is about 200 pages in length. There are 8 chapters (articles) in the book. Each chapter contains around 400 sentences comprising about 18 words each. This amounts to approximately 7200 words per chapter. To evaluate generated summaries, reference summaries of length 75 sentences are manually created for each chapter using the exact sentences from the textbook (extractive summarization). The annotators drafting the reference summaries are proficient in Hindi, have studied history at a high school level and are familiar with the course content and exam structure. In order to perform an extrinsic evaluation, we considered questions from the three most recent exam papers, from 2017-2019, and their corresponding rubrics[2]. The exams contain 3 types of questions - very short answer questions (1 mark each), short answer questions (3 marks each) and long answer questions (5 marks each) requiring 1, 3 and 5 sentences from the text respectively. There are a total of 35 questions in these exam papers. A sample very short question from the 2017 examination is as follows:

'हिंद	स्वराज'	पुस्तक	के	लेखक	का	नाम	लिखिए ।
'hind	svaraaj'	pustak	ke	lekhak	ka	naam	likhie .

Name the writer of the book 'Hind Swaraj'.

A student's response that would score one full point is as follows:

महात्मा	गाँधी	ने	'हिंद	स्वराज'	पुस्तक	लिखी ।
mahaatma	gaandhee	ne	'hind	svaraaj'	pustak	likhee .

Mahatma Gandhi wrote the book 'Hind Swaraj'.

3 Methodology

The basic idea behind the task of extractive summarization is that individual sentences in the source document are scored and ranked to extract the top n sentences as a summary. In this work, four unsupervised methods are investigated to score and rank the input document sentences. The methods used here are KLSum (Aria and Vanderwende, 2009), Luhn Summarization (Luhn, 1958), TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004). The summaries generated by these methods are evaluated and compared against each other intrinsically and extrinsically. TextRank and LexRank are graph based approaches. KLSum utilizes a probabilistic approach. Luhn uses a naive ranking algorithm based on word significance.

3.1 KLSum

Kullback–Leibler summarization (KLSum) (Aria and Vanderwende, 2009) is a probabilistic take on the extractive summarization problem. The basic idea here is to extract a summary, a set of sentences from the source document, whose unigram distribution is as close to the unigram distribution of the source document as possible. The closeness between the source and summary document distributions is determined by the KL divergence (Kullback and Leibler., 1951) $KL(D||S)$, where $D(w)$ and $S(w)$ are the unigram distributions of the word w in the source D and summary S document respectively.

$$KL(D||S) = \sum_{w} D(w)(\log(D(w)) - \log(S(w)) \quad (1)$$

The empirical unigram distribution of a document is the term frequency of words in the given document which is computed as :

$$tf_{t,d} = \frac{\textit{\# of times the term t occurs in document d}}{\textit{Total \# of terms in document d}} \quad (2)$$

Here, $tf_{t,d}$ represents term t frequency in text document d . The term frequencies are smoothened to ensure non-zero values. Mathematically, the optimization problem is defined as below:

$$S^* = \min_{S:words(S)\leq L} KL(D||S) \quad (3)$$

Here, L is the maximum number of words in the summary S. Since optimizing the above objective is exponential in the number of sentences in the source document, a greedy approach is taken. Starting with an empty summary, the summary is extracted iteratively. At each iteration, the sentence which results in minimum $KL(D||S)$ is added to the summary until the intended number of sentences is reached.

3.2 Luhn Summarization

Luhn summarization (Luhn, 1958) is a simple and naive summarization algorithm where the relative significance of each sentence in the source document is considered for selection in the summary. The basic idea exploited in this method is that an

[2]NCERT Solutions: https://byjus.com/ncert-solutions/

author of a document writing about a concept tends to repeat the same words to represent a specific notion. When such significantly repeating words are positioned relatively closer in a document, within a sentence for example, the sentence as a whole becomes significant enough to be considered in a summary.

The relative significance of each sentence is captured with the number of significant words and their physical proximity within a sentence. Each sentence is grouped into clusters beginning and ending with significant words. These first and last significant words of clusters are significantly related if the physical distance between them, intervened by insignificant words, is under a threshold. If more than one such cluster is found in a sentence, the cluster with the highest significance factor is assigned to the sentence. The sentences are then ranked relative to the other to generate the summary. Numerically, a word is considered significant if its term frequency is more than a specified threshold. The significance factor of a cluster C in a sentence is computed as follows:

$$Significance(C) = \frac{\textit{\# of significant words in C}}{\textit{\# of words in C}} \quad (4)$$

3.3 TextRank

TextRank (Mihalcea and Tarau, 2004) is a graph based approach which scores sentences in the given document based on the PageRank (Page et al., 1999) algorithm. The basic principle here is that sentences within the document recommend each other and the sentences with the highest recommendation scores are considered to be in the generated summary. This involves constructing a graphical representation of the document, $G(V, E)$, where each sentence in the document is a vertex V linked to all other vertices through edges E in the undirected graph. The edge between two vertices i and j are weighted by a similarity metric w_{ij} to capture the recommendation between sentences s_i and s_j which is calculated as follows:

$$w_{ij} = \frac{\textit{\# of } w_k | w_k \epsilon s_i, s_j}{\log(|s_i|) + \log(|s_j|)} \quad (5)$$

Here, w_k are shared tokens between sentences s_i and s_j. The PageRank algorithm is run on this constructed graph until convergence to find the importance of each vertex as per the update equation below.

$$WS(v_i) = (1-d) + d \sum_{v_j \in In(v_i)} \frac{w_{ji}}{\sum_{v_k \in Out(v_j)} w_{jk}} WS(v_j) \quad (6)$$

In the above equation, the importance score WS of vertex v_i is a function of damping factor d, incoming edge weights to a given vertex v_i, w_{ji}, and importance score $WS(v_j)$ of neighbouring vertex v_j. The vertices are ranked based on importance and the top n sentences from the document are taken as the summary.

3.4 LexRank

Like TextRank, LexRank (Erkan and Radev, 2004) is a graph based sentence scoring algorithm based on the PageRank algorithm. However, LexRank differs in the way recommendations between sentences are computed.

$$w_{ij} = \frac{\sum_{w \epsilon s_i, s_j} tf_{w,s_i} * tf_{w,s_j} * idf_w^2}{\sqrt{\sum_{w \epsilon s_i} (tf_{w,s_i} idf_w)^2} \sqrt{\sum_{w \epsilon s_j} (tf_{w,s_j} idf_w)^2}} \quad (7)$$

$$tf_{w,s_i} = \frac{\textit{\# of times the word w occurs in sentence } s_i}{\textit{\# of words in sentence } s_i} \quad (8)$$

$$idf_w = \log \frac{\textit{\# of sentences in the document}}{(1 + \textit{\# of sentences with term w})} \quad (9)$$

The similarity metric w_{ij}, between sentences s_i and s_j, is the idf-modified-cosine similarity (Erkan and Radev, 2004) computed between N dimensional vector representation of sentences.N is the number of unique terms in the document. For each word present in a sentence, the corresponding dimension in the N dimensional vector is set to the idf value of the word to construct the vector mapping of the sentence.

4 Results

The machine generated summaries are evaluated using intrinsic and extrinsic measures. Intrinsic (quantitative) evaluation uses ROUGE score (Lin, 2004) which is a recall based metric that compares similar n-grams in generated summaries against the handmade summaries. It is found that ROUGE based evaluation correlates with human based evaluation in comparing machine generated summaries with ideal summaries (Lin and Hovy, 2003). Hence, we consider ROUGE-1 and ROUGE-2 scores for

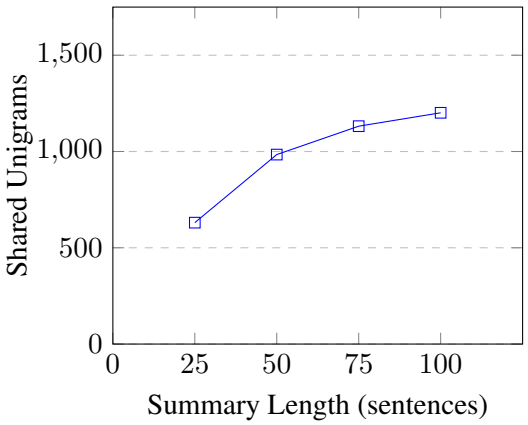


Figure 1: Overlapping unigrams vs summary length of generated summary.

this evaluation, which is the percentage of overlapping unigrams and bigrams respectively between the generated and handmade summaries.

The main idea of this work is to create a study/revision guide for students to help them understand the study material and do well on exams. Hence, the ability to answer exam questions is an indicator of a good summary. The Extrinsic (qualitative) evaluation measures how good the summary is in helping students perform well in the history exam. This is carried out by going through the summaries generated by the above mentioned algorithms and making a decision on how many points can be scored on very short and short answer questions given only the sentences in the summary. The scoring is done manually by human evaluators who refer to the examination grading rubric which is available online.

Length of the summary is an important factor to be considered when generating summaries. The challenge is to balance recall and precision, i.e. to capture as much important information as possible from the whole document while avoiding the inclusion of superfluous information. Such a summary with the right length should make for a faster and better learning experience for students. Fig. 1 shows the relationship between the length of summary, in sentences, with respect to the shared unigrams with the reference summary. While longer summaries have more overlap, the decrease in slope indicates that a decreasing percentage of added sentences match the reference. Thus, 75 sentences was selected as the model summary length.

Algorithms	ROUGE-1	ROUGE-2
LexRank	0.56	0.25
TextRank	0.72	0.44
Luhn	**0.74**	**0.45**
KLSum	0.39	0.17

Table 1: Intrinsic Evaluation: Comparison of ROUGE scores for LexRank, TextRank, Luhn and KLSum summaries compared against the reference summaries.

Algorithms	Exam scores
LexRank	40.8%
TextRank	**53.1**%
Luhn	38.8%
KLSum	46.9%
Reference Summaries	**67.3**%

Table 2: Extrinsic Evaluation: Comparison of exam scores for reference summaries, LexRank, TextRank, Luhn, and KLSum summaries.

4.1 Intrinsic Evaluation Results

Table 1 describes the ROUGE scores of different algorithm generated summaries when compared to the human generated reference summaries. We see Luhn based summaries have the highest ROUGE-1 and ROUGE-2 scores of 0.74 and 0.45 when compared to other algorithms. In this case, ROUGE-1 and ROUGE-2 follow a similar distribution.

4.2 Extrinsic Evaluation Results

The generated summaries were evaluated based on their ability to answer questions on three years' (2017-2019) history exam papers in Hindi. We compare their exam scores with the baseline, the exam scores of the hand generated reference summaries. This comparison is done by evaluators who have studied Hindi and history at a high school level while referring to the grading rubric provided by the CBSE board. It is important to note that the full text is sufficient to answer all of the exam questions scoring 100%. The reference summaries scored 67.3% outperforming summaries of TextRank scoring 53.1%, LexRank scoring 40.8%, Luhn scoring 38.8% and KLSum scoring 46.9% as shown in Table 2.

5 Discussion

When evaluated extrinsically on question answering ability, we see that human generated reference

summaries are able to score better on exam questions when compared to machine generated summaries. Among the unsupervised approaches, TextRank scores the most on exam questions, 53.1%. Since TextRank is able to answer approximately 80% of exam questions that the reference summaries answer, we believe that note generation by TextRank provides a good supplementary study tool for students.

We observed the impact of Hindi on the ROUGE metric. The presence of stop words, ambiguous pronouns and other commonly used connecting terms in Hindi artificially raise the n-gram overlap without adding useful information. For example, consider the two sentences below:

नेहरू ने गाँधी से सत्याग्रह का महत्व को समझा ।
neharoo ne gaandhee se satyaagrah ka mahatv ko samajha .
Nehru learnt the importance of non-violence from Gandhi.

गाँधी ने अंग्रेज़ को भारत से निकलने का आग्रह किया ।
gaandhee ne angrez ko bhaarat se nikalane ka aagrah kiya .
Gandhi urged the British to leave India.

The two sentences have completely different meanings, sharing only subject, Gandhi in common. The English sentences have two unigrams in common, 'Gandhi' and 'the' out of a total of thirteen unique unigrams, approximately a 15% overlap. On the other hand, the Hindi sentences have a total of five unigrams in common out of a total of fourteen unique unigrams, approximately a 36% overlap. This aspect of the Hindi language, with an abundance of connecting terms, would also raise the ROUGE metric of sentences which need not convey useful information.

We reevaluated the importance of the ROUGE score for the chosen task. We notice that a good ROUGE score is not a good indicator of a summary's ability to serve as a study aid. This is evident from the extrinsic evaluation. Luhn summarization, which has the highest ROUGE-1 score (0.74), performs poorly on the question answering task scoring only 38.8%. Conversely, KLSum having the lowest ROUGE-1 score (0.39) performs better than Luhn summarization extrinsically, obtaining approximately 47%. This relationship between the ROUGE and exam scores of the summaries can be confirmed by the Spearman's rho and Kendall's tau coefficients (Yue et al., 2002) , which are -0.4 and -0.33 respectively. The negative coefficients indicate a weak correlation between the summary's ROUGE score and its question answering ability. This shows that, in addition to ROUGE, it is important to formulate evaluation mechanisms that align with chosen application to evaluate machine generated summaries.

We noticed that machine generated summaries have sentences with ambiguous subjects. While the algorithms may identify an important fact, it cannot attribute it to a subject. Consider the following sentence:

स्कूल का प्रिंसिपल एक कोलोन था । उसने लड़की को स्कूल से निकाल दिया ।
scool ka prinsipal ek kolon tha . Usne ladakee ko scool se nikaal diya .
The school's principal was a colonist. He expelled the girl from the school.

When the machine generated summary contains only the second sentence it is able to answer the question "What caused the school rebellion?" (expelling the girl from school) but cannot identify the subject (school's principal) who carried out the action without the preceding sentence. This is a structure we see often in Hindi where one sentence in English corresponds to two in Hindi. In the English version of the text, the fact is stated as

"The principal, also a colon, expelled her."

As the input text documents to the models were not pre-processed, we observed models treating the same entity differently. For instance, the tokens Gandhi and Gandhi-ji. The addition of an honorific suffix 'ji' results in both terms being treated as different. Since the rule of removing the suffix 'ji' applies only to proper nouns, we cannot generalize this as a stemmer rule.

We believe that a TextRank based summarization tool would prove effective for other subjects whose exam questions test factual knowledge like Geography or Biology. However, further testing is required before its portability can be validated. Also, we believe the project would benefit from an Entity Recognizer, as a pre-processing step, to solve both ambiguous subjects problems and the ambiguity caused by the honorific suffixes in the summaries. Nevertheless, we believe that this project represents a step in the right direction towards providing a note generation tool for students in Hindi medium schools.

References

Haghighi Aria and Lucy Vanderwende. 2009. Exploring content models for multi-document summariza-

tion. In *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Navneet Garg, Vishal Goyal, and Suman Preet. 2012. Rule based hindi part of speech tagger. In *Proceedings of COLING 2012: Demonstration Papers*, pages 163–174.

Manisha Gupta and Naresh Kumar Garg. 2016. Text summarization of hindi documents using rule based approach. In *2016 international conference on micro-electronics and telecommunication engineering (ICMETE)*, pages 366–370. IEEE.

Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. pages 79–86.

K Vimal Kumar, Divakar Yadav, and Arun Sharma. 2015. Graph based technique for hindi text summarization. In *Information Systems Design and Intelligent Applications*, pages 301–310. Springer.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

NCERT. 2018-2019. *Bharat Aur Samkalin Vishwa-2*. National Council of Education Research and Training. ISBN: 81-7450-712-4.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Alok Ranjan Pal and Diganta Saha. 2014. An approach to automatic text summarization using wordnet. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 1169–1173. IEEE.

Snigdha Paul, Nisheeth Joshi, and Iti Mathur. 2013. Development of a hindi lemmatizer. *arXiv preprint arXiv:1305.6211*.

Pradeepika Verma, Sukomal Pal, and Hari Om. 2019. A comparative analysis on hindi and english extractive text summarization. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(3):1–39.

Sakshee Vijay, Vartika Rai, Sorabh Gupta, Anshuman Vijayvargia, and Dipti Misra Sharma. 2017. Extractive text summarisation in hindi. In *2017 International Conference on Asian Language Processing (IALP)*, pages 318–321. IEEE.

Sheng Yue, Paul Pilon, and George Cavadias. 2002. Power of the mann–kendall and spearman's rho tests for detecting monotonic trends in hydrological series. *Journal of hydrology*, 259(1-4):254–271.

Unbiasing Review Ratings with Tendency based Collaborative Filtering

Pranshi Yadav*
IIIT, Hyderabad
pranshiyadav@gmail.com

Priya Yadav*
JSSATE, Noida
priya.yadav252@gmail.com

Pegah Nokhiz
University of Utah
pnokhiz@cs.utah.edu

Vivek Gupta
University of Utah
vgupta@cs.utah.edu

Abstract

User-generated contents' score-based prediction and item recommendation has become an inseparable part of the online recommendation systems. The ratings allow people to express their opinions and may affect the market value of items and consumer confidence in e-commerce decisions. A major problem with the models designed for user review prediction is that they unknowingly neglect the rating bias occurring due to personal user bias preferences. We propose a tendency-based approach that models the user and item tendency for score prediction along with text review analysis with respect to ratings.

1 Introduction

Our society is increasingly relying on the digitized, aggregated opinions of others, which may be biased and easily manipulated while making decisions. Online reviews typically have a distribution of opinions, i.e., many are extremely positive/negative, and a few are moderate opinions.

However, numerous online reviews suffer from the rating bias problem. The opinions of individual reviewers may be affected when a person allows their preformed personal bias to affect the evaluation of another. Some users are very generous and do not rate an item with less than a 3 or 4 (on a scale of 5), thus introducing a positive bias in the scores. In contrast, some users do not go beyond a 1 or 2, thus introducing a negative bias. Hence, the reviews can affect the product's market positively or negatively, regardless of the actual product performance. The rating bias problem was earlier studied as user bias problem by Adomavicius et al. (2014); Guo and Dunson (2015); Abdollahpouri et al. (2017, 2019); Abdollahpouri (2019).

In this paper, we focus on extending a simple normalization approach by Wadbude et al. (2018), as shown in Figure 1, to mitigate user or item bias.

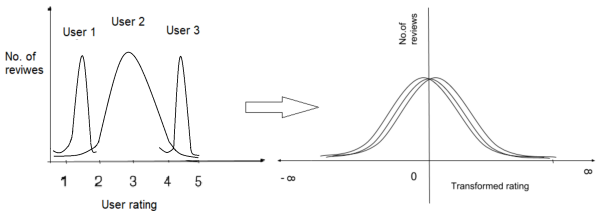


Figure 1: The user item bias problem (Wadbude et al., 2018)

Instead of using mean and standard deviation-based review rating normalization over all reviews (Wadbude et al., 2018), we perform user/item-specific review rating normalization in order to identify user/item-specific bias in ratings. That is, we will use an intuitive tendency based approach, a well-studied concept in the recommender systems literature (Sreepada et al., 2018; Cacheda et al., 2011), that estimates the bias for all user-item pairs based on user and item means/tendencies and predicts the unbiased score. Furthermore, our approach can also predict the rating of new review pairs of users and items, assuming the users or the items have a history of reviews with their corresponding ratings (i.e., no cold start setting). Therefore, we refer to Wadbude et al. (2018) for the problem introduction and Sreepada et al. (2018); Cacheda et al. (2011) for coming up with a tendency-based solution for bias mitigation and new user-item rating prediction.

The contributions in this paper are the following:

- A simple yet effective framework, inspired by Sreepada et al. (2018); Cacheda et al. (2011), based on user and item means and tendencies to underscore and mitigate the rating bias problem in reviews. Using this framework, we obtain unbiased ratings for user-item pairs.
- Furthermore, we extended the above framework to predict ratings for a new user-item pair, provided that the user or item is not novel

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 50–56

and has prior feedback rating (not a cold start setting). To achieve this, we come up with case-dependent reverse estimation functions.

- Through extensive experiments on publicly available datasets, we show that our approach not only helps in mitigating users' rating bias and predicting ratings for new user-item pairs, but also in detecting the exceedingly biased users/items (i.e., outliers in the review dataset) and a better alignment of predicted ratings with review text sentiments than the original scores.

In Section (1), we provided a brief introduction to the problem statement. The remaining parts of the paper are organized as follows: in Section (2) we explain our approach to predict ratings for new user-item pairs. In Section (3), we explain the experimental details. We then move on to our results and analysis in Section (4). Next, we discuss the related work in (5), followed by conclusions and future work in Section (6). We have also released the source code along with the paper.[1]

2 Rating Prediction for New User-items

We propose a user-specific statistical mapping based on user and item tendency for rating-bias removal by normalizing each review score with respect to the user and item tendencies. User tendency means whether a user tends to rate every item positively or negatively in general. Item tendency refers to whether the underlying population of users considers a specific item particularly good or bad. This is different from comparing the mean rating of the item to the global mean. The goal is to see if the item stands out among all the products rated by the user (Cacheda et al., 2011).

Let $r(c_u, p_i)$ represent the review score of user (customer) c_u for item p_i. As explained in Cacheda et al. (2011); Sreepada et al. (2018), we calculate the unbiased rating $\widehat{r_{c_u,p_i}}$ from the available ratings of user c_u and item p_i, and the predicted rating (with bias) $r(c_k, p_m)$ for a new review of user c_k for item p_m, i.e., a new user-item pair as follows:

1. Get the mean user rating $\overline{r_u}(c_u) = \frac{1}{n_p}\sum_{j=1}^{n_p} r(c_u, p_j)$ using all the reviews given by user c_u for each user. Similarly, for each item calculate the mean item rating $\overline{r_i}(p_i) = \frac{1}{n_c}\sum_{j=1}^{n_c} r(c_j, p_i)$ using the ratings given by all the users who rated that item. Here, n_p is the number of items reviewed by user c_u, and n_c is the number of reviews for item p_i.

2. Now, for every user c_u, store the user tendency $\tau_u(c_u) = \frac{1}{n_p}\sum_{i=1}^{n_p}(r(c_u, p_i) - \overline{r_i}(p_i))$ using all the reviews provided by the user c_u. Similarly, for every item p_i, we calculate the item tendency $\tau_i(p_i) = \frac{1}{n_c}\sum_{u=1}^{n_c}(r(c_u, p_i) - \overline{r_u}(c_u))$ using all the review scores given for the item p_i. Here also, n_p is the number of items reviewed by user c_u, and n_c is the number of reviews for item p_i.

   For simplicity, for rest of the paper we represent rating $r(c_u, p_i)$ as r_{ui}, user mean $\overline{r_u}(c_u)$ as $\overline{r_u}$, item mean $\overline{r_i}(p_i)$ as $\overline{r_i}$, user tendency $\tau_u(c_u)$ as τ_u, and item tendency $\tau_i(p_i)$ as τ_i. Furthermore, we will also refer to n_p as n_u, and n_c as n_i for rest of the paper.

3. To calculate the unbiased ratings $\widehat{r_{c_u,p_i}}$ for a given user-item pair (c_u, p_i), we first calculate the user and item means (i.e., $\overline{r_u}$ and $\overline{r_i}$, respectively) and user and item tendencies (i.e., τ_u and τ_i, respectively). Based on the comparison of the user mean and item mean and the sign of the user tendency and item tendency, different cases are defined for the calculation of the unbiased rating $\widehat{r_{c_u,p_i}}$ for a given user-item pair (c_u, p_i) as shown in Table 1. The β (between 0 and 1) in Table 1 is a hyper-parameter that controls the contribution of the item mean user tendency and user mean item tendency for unbiased rating calculation. For all our experiments we set β to the standard value of 0.5. For simplicity, we represent unbiased rating $\widehat{r_{c_u,p_i}}$ as $\widehat{r_{ui}}$ for the rest of the paper.

Case	Unbias Function ($\widehat{r_{ui}}$)
1. $\tau_u > 0, \tau_i > 0$	$\max(\overline{r_u} + \tau_i, \overline{r_i} + \tau_u)$
2. $\tau_u < 0, \tau_i < 0$	$\min(\overline{r_u} + \tau_i, \overline{r_i} + \tau_u)$
3. $\tau_u < 0, \tau_i > 0, \overline{r_u} < \overline{r_i}$	$\min(\max(\overline{r_u}, (\overline{r_i} + \tau_u)\beta + (\overline{r_u} + \tau_i)(1-\beta)), \overline{r_i})$
4. $\tau_u < 0, \tau_i > 0, \overline{r_u} > \overline{r_i}$	$\overline{r_i}\beta + \overline{r_u}(1-\beta)$
5. $\tau_u > 0, \tau_i < 0, \overline{r_u} > \overline{r_i}$	$\min(\max(\overline{r_i}, (\overline{r_u} + \tau_i)\beta + (\overline{r_i} + \tau_u)(1-\beta)), \overline{r_u})$
6. $\tau_u > 0, \tau_i < 0, \overline{r_u} < \overline{r_i}$	$\overline{r_u}\beta + \overline{r_i}(1-\beta)$

Table 1: Unbiasing functions (Sreepada et al., 2018)

4. Finally, we use the previous bias functions to come up with reverse estimation functions and recover the original biased rating by these

[1] https://github.com/pranshiyadav06/review-bias-normalization

functions. Reverse estimation functions are functions that restore the bias and add it to the unbiased rating for observed user/item rating predictions. We predict a review rating $r(c_k, p_m)$ (i.e., r_{km}) for the new review of user c_k for item p_m, a new user-item pair, using these reverse estimation functions. These functions are not universal and depend on user and item parameters (i.e., user/item means and tendencies). Table 2 lists the reverse functions for all the possible cases that may exist. To account for space constraints, the detailed derivations of reverse estimation functions are mentioned in appendix.[2] The ratings obtained by using the reverse estimation functions can now be compared with the given ratings (gold ratings) to evaluate the effectiveness of our tendency-based approach.

3 Experimental Details

Equations derived in Section 2, can be used to calculate the unbiased ratings for a user-item pair and predicting the ratings for a new user-item pair. We perform extensive experiments with ratings as well as the review texts associated with these ratings to evaluate our method.

3.1 Enough Rating Assumption

To empirically test this approach using real-world datasets, we need to calculate adequate invariant estimates of user/item means and tendencies from the rated data. To do so, we need to make the following assumption: The prediction example is in an online setting. Thus, the distribution means and tendencies (both users/items) will not change significantly when we predict ratings for new user-item pairs. This helps us decide the case on the estimated mean and tendency from the given labeled ratings. Similarly, the case for reverse estimation functions will be decided from prior data's means and tendencies.[3]

3.2 Dataset Details

We tested our approach on the electronics category of the SNAP Amazon e-Commerce Reviews dataset (McAuley and Leskovec, 2013). We also tested our approach on the Amazon Fine Food Reviews dataset. All the ratings in both datasets are integers from 1 to 5. In addition to review ratings, the Amazon Fine Food Reviews dataset provides the corresponding review texts and the helpfulness score of the reviews' texts. The review text could be used to obtain an alignment pattern between the differences in text reviews and their ratings. These review scores are used to get an adequate invariant estimate of each of the user/item means and tendencies from the review ratings.

Furthermore, in order to hold the assumption of an online setting, we need to have a substantial number of ratings for all users and items. To do so, we removed the items/users that have fewer ratings than a certain threshold, i.e., if the count of ratings was below 15 for Electronics and 12 for Fine Food for each user/item in both datasets. Table 3 shows the number of labeled (where the rating is provided) and unlabeled (where the rating is hidden) reviews used in the experiments after pruning. It also lists the number of users and items available in the unlabeled set. In order to evaluate the model, we have calculated the average difference in the predicted biased score and original review score for each user and item.

3.3 Experiment Procedure

Depending on the case for the mean and tendency explained in Section 2, we calculated the unbiased review score from Table 1 for the new user-item pair in the unlabeled set. To predict the review score that the user might assign to the item, we used the reverse estimation functions (Table 2, Section 2). Note that these functions will not take the minor change in the mean and tendency value that might occur due to a new user-item pair into consideration.

4 Results and Analysis

We answer the following questions through the experiments:

- Can we use a tendency-based approach to identify and remove the bias in the review ratings?
- Can we use these tendencies to predict the rating that a user may assign to an item?[4]

[2] Derivation of reverse estimation functions appendix: https://github.com/pranshiyadav06/review-bias-normalization/blob/master/Appendix.pdf

[3] One can employ a verification step after updating the means and tendencies to validate this assumption. It could be used to update the prior with the posterior distribution for modeling means and tendencies with fully Bayesian updates.

[4] The user or item is not entirely fresh and has previous reviews associated with them, i.e., no cold start setting.

Case	Sub-case	Reverse tendency estimation function
1.$\tau_u > 0, \tau_i > 0$	I. $\widehat{r_{ui}} = \overline{r_u} + \tau_i(new)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[\widehat{r_{ui}}(n_u+1)(n_i+1) - (n_i+1)(n_u \times \tau_i) - (n_i \times n_u \times \overline{r_u})]$
	II. $\widehat{r_{ui}} = \overline{r_i} + \tau_u(new)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[\widehat{r_{ui}}(n_u+1)(n_i+1) - (n_u+1)(n_i \times \tau_u) - (n_i \times n_u \times \overline{r_i})]$
2.$\tau_u < 0, \tau_i < 0$	I. $\widehat{r_{ui}} = \overline{r_u} + \tau_i(new)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[\widehat{r_{ui}}(n_u+1)(n_i+1) - (n_i+1)(n_u \times \tau_i) - (n_i \times n_u \times \overline{r_u})]$
	II.$\widehat{r_{ui}} = \overline{r_i} + \tau_u(new)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[\widehat{r_{ui}}(n_u+1)(n_i+1) - (n_u+1)(n_i \times \tau_u) - (n_i \times n_u \times \overline{r_i})]$
3.$\tau_u < 0$, $\tau_i > 0, \overline{r_u} < \overline{r_i}$	I. $\widehat{r_{ui}} = \overline{r_{u(final)}}$	$r_{ui(new)} = (n_i+1)\widehat{r_{ui}} - (\overline{r_u} \times n_i)$
	II. $\widehat{r_{ui}} = (\overline{r_i} + \tau_u)\beta + (\overline{r_u} + \tau_i)(1-\beta)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[(n_u+1)(n_i+1)\widehat{r_{ui}} - (n_u)(\overline{r_u} \times n_i)(1-\beta) - (n_i+1)(1-\beta)(n_u \times \tau_i) - \beta(n_i)(\overline{r_i} \times n_u) - \beta(n_u+1)(n_i \times \tau_u)]$
	III. $\widehat{r_{ui}} = \overline{r_{i(final)}}$	$r_{ui(new)} = (n_u+1)\widehat{r_{ui}} - (\overline{r_i} \times n_u)$
4.$\tau_u < 0, \tau_i > 0$, $\overline{r_u} > \overline{r_i}$	$\widehat{r_{ui}} = \overline{r_i}\beta + \overline{r_u}(1-\beta)$	$r_{ui(new)} = [\frac{1}{n_u(1-\beta)+1+\beta n_i}] \times [(n_u+1)(n_i+1)\widehat{r_{ui}} - \beta(n_i+1)(\overline{r_i} \times n_u) - (1-\beta)(n_u+1)(\overline{r_u} \times n_i)]$
5.$\tau_u > 0$, $\tau_i < 0, \overline{r_u} > \overline{r_i}$	I. $\widehat{r_{ui}} = \overline{r_i(final)}$	$r_{ui(new)} = (n_u+1)\widehat{r_{ui}} - (\overline{r_i} \times n_u)$
	II.$\widehat{r_{ui}} = (\overline{r_u} + \tau_i)\beta + (\overline{r_i} + \tau_u)(1-\beta)$	$r_{ui(new)} = (\frac{1}{n_i+n_u+1})[(n_u+1)(n_i+1)\widehat{r_{ui}} - (n_i)(\overline{r_i} \times n_u)(1-\beta) - (n_u+1)(1-\beta)(n_i \times \tau_u) - \beta(n_u)(\overline{r_u} \times n_i) - \beta(n_i+1)(n_u \times \tau_i)]$
	III. $\widehat{r_{ui}} = \overline{r_u(final)}$	$r_{ui(new)} = (n_i+1)\widehat{r_{ui}} - (\overline{r_u} \times n_i)$
6.$\tau_u > 0, \tau_i < 0$, $\overline{r_u} < \overline{r_i}$	$\widehat{r_{ui}} = \overline{r_u}\beta + \overline{r_i}(1-\beta)$	$r_{ui(new)} = [\frac{1}{n_i(1-\beta)+1+\beta n_u}] \times [(n_u+1)(n_i+1)\widehat{r_{ui}} - \beta(n_u+1)(\overline{r_u} \times n_i) - (1-\beta)(n_i+1)(\overline{r_i} \times n_u)]$

Table 2: Reverse estimation functions to predict review ratings (with bias) for a new user-item pair

Dataset	#Label	#Unlabel	#users	#items
Electronics	303K	75K	17486	18914
Fine Food	75K	18K	4904	3978

Table 3: Dataset statistics (after pruning)

- Can we identify anomalous ratings given the fact that a biased user can rate highly positively or negatively irrespective of the item quality?
- Can we find an alignment pattern between the difference in the reviews written for an item, the given rating, and the predicted rating?

4.1 Unbiased Ratings

We observed that for most of the user-item pairs, the unbiased review scores lie in the upper range of the rating spectrum (around 4), as shown in Figures 2 and 3. This shows that most of the people do not consistently assign poor review scores. This could also be due to the fact that more rated items are of good quality; thus, the users provide satisfactory reviews for these items.

4.2 Predicted Ratings

For the unlabeled set (hidden ratings) we calculated the predicted ratings by using the reverse estimation function (Table 2, Section 2). We then compared our predicted rating estimations with the true hidden ratings by using standard metrics such as the mean absolute error, mean squared error, and root mean squared error. Table 4 shows the error value returned on the unlabeled set for both datasets. On average, the predicted value differs from the actual value, which was assigned by the user within a range of 0.75. Therefore, our approach can accurately predict the rating of a new user-item pair.

Dataset	MAE	MSE	RMSE
Electronics	0.75	1.07	1.03
Fine Food	0.57	0.78	0.88

Table 4: Prediction rating on the unlabeled dataset

We also plotted the distribution of the difference in the predicted and the actual ratings. Figures 4 and 5 show the variation in the difference between the predicted review ratings and the actual ratings for items and users (in Electronics), respectively. The distribution is close to a normal with a mean (and peak at) zero and a standard deviation in a range of < 1.0. A similar distribution was observed for Fine Food reviews, as shown in Figures 6 and 7. Hence, we conclude that most predicted values lie within a low-margin error range of review ratings.

Table 5 represents the number of users and items with a certain average error (difference in predicted and actual review score) in various intervals of error. This shows that the majority of predicted review scores have an acceptable error of < 1.0.

4.3 Outliers

As shown in Figures 4 and 5, there are a few user-item pairs whose average error between predicted and actual ratings is substantially high. Neverthe-

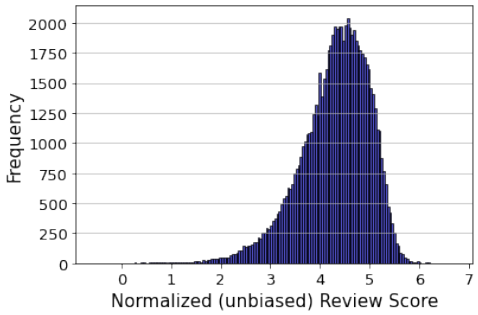


Figure 2: Unbiased rating distribution of Amazon Electronics

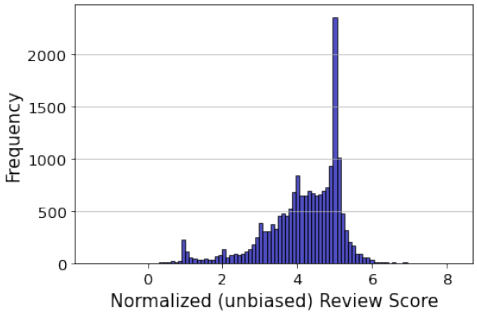


Figure 3: Unbiased rating distribution of Amazon Fine Food Reviews

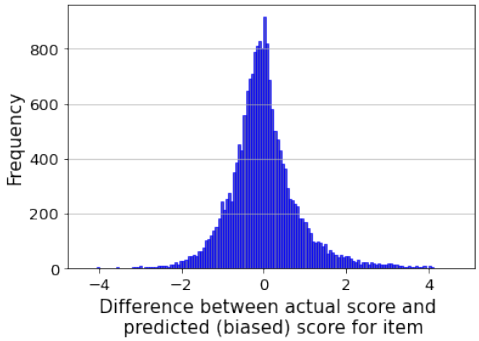


Figure 4: Average difference in predicted and Actual Review Score for items for Amazon Electronics

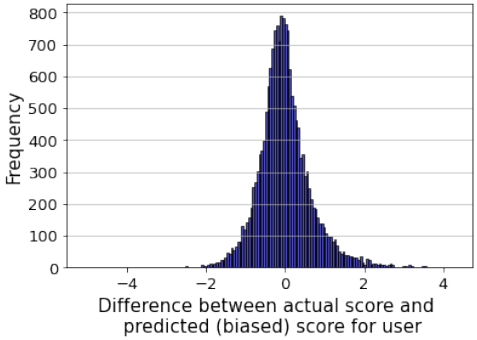


Figure 5: Average difference in predicted and Actual Review Score for users for Amazon Electronics

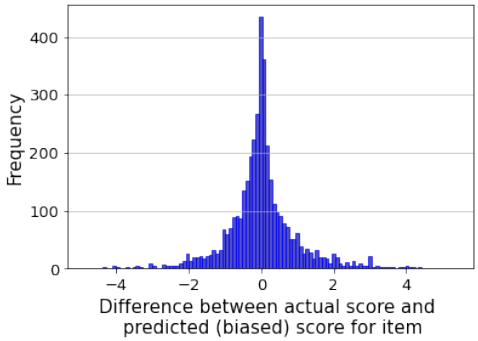


Figure 6: Average difference in predicted and Actual Review Score for items for Amazon Fine Food Review

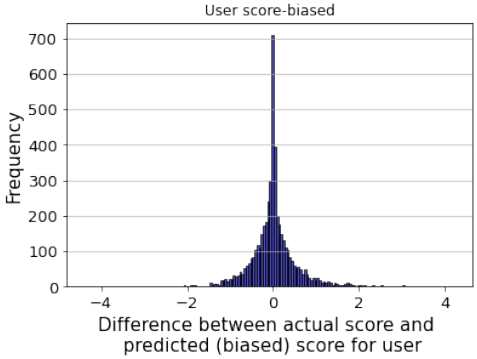


Figure 7: Average difference in predicted and Actual Review Score for users for Amazon Fine Food Review

Interval	**#users /% of total (Electronic | Food)**	**#items/% of total (Electronics | Food)**
- 0.2 – 0.2	5252/30 | 2366/48	5175/27 | 1357/34
- 0.5 – 0.5	11057/63 | 3657/75	10707/57 | 2324/58
- 1.0 – 1.0	15353/88 | 4468/91	15329/81 | 3139/80
> 2 or < 2	286/1.6 | 79/1.6	788/4 | 289/7

Table 5: Number of users and items for various error ranges

less, the number of such pairs is insignificant compared to the total number of pairs. Only 1.6% of users and 4% of items have such outliers (refer to Table 5). A possible reason behind these outliers is that a positively biased user can get a defective item and rate it with an extremely poor score and vice-versa. Table 6 shows some of the examples where the rating is extremely good for negatively biased (negative tendency) users and items, and extremely bad for positively biased users and items.

item / user	**Bias Score**	**Unbias Score**	**Predicted Score**	**Diff**
I1 / U1	5.0	2.68	2.75	-2.25
I2 / U2	2.0	3.68	4.06	2.06
I3 / U3	1.0	3.43	3.50	2.50
I4 / U4	1.0	5.35	5.04	4.04
I5 / U5	5.0	-0.55	-0.06	-5.06

Table 6: User-item outlier pairs' predicted ratings

4.4 Review Text Analysis

Table 7 shows that there are some user-item pairs whose review text summary is ambiguous, and the predicted score from our approach is closer to the sentiment of the text review rather than the labeled review score provided in the data. In addition, this ambiguity results in a possible range of review ratings (positive or negative) that can be assigned to the text. Therefore, the labeled rating would be the most extreme value if the user-item is biased. An explanation for this could be the presence of inherent bias in the user or item. Writing a standalone textual review mitigates the user bias compared to merely assigning a singular rating as feedback.

Summary	Orignal Score	Predict Score
doe a good job but no miracl	4.0	3.03
wonder sound high qualiti stiff cord	4.0	4.67
mix impress	3.0	3.89
super slick look use of magnet be	3.0	3.83
d solid mid level nikon dslr	4.0	4.93

Table 7: Review text analysis

Note that the review rating prediction does not take into account the sentiments expressed in the review text. That is, the reverse tendency estimation functions do not take into account the sentiment scores from the text of the review, and here, we are just using the review text to crosscheck and contrast the sentiments in the text with the original and predicted ratings.

5 Related Work

Most of the earlier approaches in rating prediction fall into three categories: a) Sentiment Analysis, b) Recommendation Systems (with/out collaborative filtering), and c) A hybrid of both. In sentiment analysis papers, the review text is used to predict the rating. This approach completely ignores the user/item meta-data, which is useful for rating bias estimation. Multiple instances of works fall in this category, but only a few are closer to our approach, e.g., Wang et al. (2016); Lei et al. (2017).

In the recommendation setting, rating predictions are treated as a matrix completion problems where the similarity between users is utilized for recommending new items. For example, if user a is similar to user b, one can assume that the rating of both users would be similar for a given item. In this setting, earlier work ignores the rating bias in users and items' tendencies. Another line of similar work, such as Musat et al. (2013); Sreepada et al. (2018); Cacheda et al. (2011) uses tendency based collaborative filtering for rating prediction. Similar to before, this line of work either ignores or does not thoroughly analyze the rating bias problem.

There are some attempts, such as Ling et al. (2014); Du et al. (2017); Jakob et al. (2009); Pero and Horvath (2013); Rao et al., which use a hybrid model for both personalized sentiment prediction and recommendation by combining information from both review and text; however, they too have not acknowledged the rating bias problem in detail. The rating bias problem was earlier studied as user bias problem by Adomavicius et al. (2014); Guo and Dunson (2015); Abdollahpouri et al. (2017, 2019); Abdollahpouri (2019). However, the most relevant work to our approach is Wadbude et al. (2018), which acknowledges the rating bias problem using simple mean/standard deviation-based normalization schemes. Although, in our paper, we use simple yet effective tendency based collaborative filtering instead of a standard normalization.

6 Conclusion

In this paper, a tendency-based approach that models the user and item tendency for rating prediction is proposed and tested on standard review datasets. To do so, we use the existing established work on tendency-based collaborative filtering for obtaining reverse estimation functions. Our method can successfully mitigate user rating bias and can also help in detecting outliers/anomalies in ratings and reviews' texts. We will further extend this model by analyzing tendencies and score predictions based on the users' demographics and items' categories.

Currently, our model cannot handle the cold start problem and require a sufficient history of earlier ratings for users/products. Thus, our model cannot predict ratings for completely new users and items. We intend to examine this problem in the future extensions of the model. Moreover, despite the bias in the text and score rating, we noted that bias is more inherent while rating a product. As shown in Table 7, we see that text can be ambiguous or neutral while the review score has a substantial amount of bias in it. We plan to extend the proposed model to take into account the positive and negative text-based sentiments along with the scores for a more accurate bias modeling. Thus, using the review text sentiments is a possible future direction. One could also use the sentiments as a means to compare with techniques that detect bias from the text.

References

Himan Abdollahpouri. 2019. Popularity bias in ranking and recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 529–530. ACM.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 42–46. ACM.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555*.

Gediminas Adomavicius, Jesse C Bockstedt, Curley Shawn, and Jingjing Zhang. 2014. De-biasing user preference ratings in recommender systems. In *Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2014, Colocated with ACM Conference on Recommender Systems, RecSys 2014*, pages 2–9. CEUR-WS.

Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):1–33.

Yali Du, Chang Xu, and Dacheng Tao. 2017. Privileged matrix factorization for collaborative filtering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 1610–1616. IJCAI.

Fangjian Guo and David B Dunson. 2015. Uncovering systematic bias in ratings across categories: A bayesian approach. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 317–320.

Niklas Jakob, Stefan Hagen Weber, Mark Christoph Muller, and Iryna Gurevych. 2009. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. CIKM.

Zheng Lei, Vahid R. Noroozi, , and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM International Conference on Web Search and Data*, pages 425–434. ACM.

Guang Ling, Michael R. Lyu, , and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 105–112. ACM.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Claudiu-Cristian Musat, Yizhong Liang, and Boi Faltings. 2013. Recommendation using textual opinions. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 2684–2690. IJCAI.

Stefan Pero and Tomas Horvath. 2013. Opinion-driven matrix factorization for rating prediction. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 1–13. UMAP.

K Yogeswara Rao, GSN Murthy, and S Adinarayana. Product recommendation system from users reviews using sentiment analysis. *International Journal of Computer Applications*, 975:8887.

Rama Syamala Sreepada, Bidyut Kr Patra, Avijit Chakrabarty, and Satyadev Chandak. 2018. Revisiting tendency based collaborative filtering for personalized recommendations. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 230–239. ACM.

Rahul Wadbude, Vivek Gupta, Dheeraj Mekala, and Harish Karnick. 2018. User bias removal in review score prediction. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 175–179. ACM.

Bingkun Wang, Yongfeng Huang, and Xing Li. 2016. Combining review text content and reviewer-item rating matrix to predict review rating. In *Computational Intelligence and Neuroscience*. Hindawi.

Building a Part-of-Speech Tagged Corpus for Drenjongke (Bhutia)

Mana Ashida
Tokyo Metropolitan University
ashida-mana@ed.tmu.ac.jp

Seunghun J. Lee
International Christian University
University of Venda
seunghun@icu.ac.jp

Kunzang Namgyal
Nar Bahadur Bhandari Degree College
kunzang49348@gmail.com

Abstract

This research paper reports on the generation of the first Drenjongke corpus based on texts taken from a phrase book for beginners, written in the Tibetan script. A corpus of sentences was created after correcting errors in the text scanned through optical character reading (OCR). A total of 34 Part-of-Speech (PoS) tags were defined based on manual annotation performed by the three authors, one of whom is a native speaker of Drenjongke. The first corpus of the Drenjongke language comprises 275 sentences and 1379 tokens, which we plan to expand with other materials to promote further studies of this language.

1 Introduction

According to Joshi et al. (2020), most of the world's languages cannot benefit from the state-of-the-art methods of Natural Language Processing (NLP) due to a lack of resources. Given that the need for language technologies is equally distributed among speakers of each language, the investigation of the construction of language resources is necessary. This paper reports how the first Drenjongke machine-readable labeled data were developed.

Drenjongke (Bhutia[1]) is a Tibeto-Burman language with the status of one of the official languages in the state of Sikkim, India. Yliniemi (2019) estimates that the number of Drenjongke speakers is 25,000-30,000 and that Drenjongke falls into one of the categories of "vulnerable," "definitely endangered," and "severely endangered" within UNESCO's Language Vitality and Endangerment framework, with a rapid loss of native speakers due to the lack of economic value in speaking Drenjongke. The low economic value is due to the widespread use of Nepali in Sikkim. Most official and private business matters are conducted in Nepali or English. As a result of community-driven revitalization efforts[2], written materials in Drenjongke are being produced, including but not limited to dictionaries, folk songs, and novels. An archive was created resulting from a recent project with Drenjongke community members: "Phonetics, Phonology and New Orthographies in Roman and Indigenous Script: Helping Native Language Communities in the Himalayas (2017-2020)."[3] The archive contains spoken materials, including conversations and nursery rhymes.[4] However, the focus of the archive was speech recordings. Thus, machine-readable texts were not included.

The goal of our project is to build the first digitized resource of Drenjongke so as to serve as a basis for expanding NLP resources on Drenjongke by suggesting a set of PoS tags that can be used for processing other Drenjongke texts in the future. Moreover, we provide annotated text materials that would promote future computational research.

2 Related Work in Other Tibeto-Burman Languages

Tibeto-Burman languages share many characteristics. From NLP perspectives, some similarities include 1) orthographic systems that are not based on the Roman alphabet, and 2) agglutinative morphology that creates difficulty in determining the word boundaries often used in NLP systems targeting Indo-European languages. Even so, annotated text corpora in Burmese (Myanmar), Dzongkha

[1] Drenjongke is known as Bhutia, Sikkimese, and also Lhoke. Community members prefer the terminology Drenjongke over others. Bhutia is the official name accepted in India (Yliniemi, 2019).

[2] A native speaker shared with us the observation that the latest number of speakers may reach as high as 40,000 to 50,000 with revitalization efforts.

[3] https://phophono.aa-ken.jp/

[4] The nursery rhymes are also accessible at the following YouTube channel https://www.youtube.com/channel/UC90NszbLUgoOw7ZLEUHJ3WA/ along with their lyrics in both Drenjongke and English.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 57–63

(Bhutan), and Classical and Modern Tibetan exist.

The largest existing Burmese corpus with annotations is the "Burmese (Myanmar) Treebank of Asian Language Treebank Project" (Ding et al., 2019). This corpus contains 20,000 sentences that are morphologically annotated with the annotation scheme "nova," in which *n* stands for nouns, *v* for verbs, *a* for adjectives, and *o* for others. This categorization was created for low-resource, highly analytic languages. Subsequently, Ding et al. (2020) propose the conversion procedure of the "nova" tag to the Universal Part-of-Speech (PoS) tagset.[5]

Corpora as well as PoS taggers in Classical Tibetan[6] have been developed as part of the project "Tibetan in Digital Communication (2012-2015)" hosted at SOAS.[7] The project has annotated four documents from Tibetan classical literature.[8] As for Modern Tibetan, Liu and Congjun (2018) report a collection of Tibetan text corpora (CTTC) containing 52,041 PoS-tagged sentences, which have been used in Machine Translation and Tibetan syntactic parsing.

Dzongkha also has several NLP resources, including a corpus and tokenizers. Norbu et al. (2010) propose a word segmentation method applied to Dzongkha using the dictionary-based maximal matching algorithm. Chungku et al. (2010) describe the corpus building process for Dzongkha, in which they prepared 66 PoS tags for annotation.

We found that these NLP resources on Tibeto-Burman languages often lack uniformed annotation guidelines as pointed out in Hill and List (2017). As such, when building corpora in a different low-resource Tibeto-Burman language, namely Drenjongke, information about the process of building of existing corpora was not directly transferable.

3 Corpus Construction

Our corpus contains 275 sentences and 1379 tokens taken from Bhutia Phrase Book (Drenjongpo, 2017). The corpus has two parts: the sentence level, containing (a) sentence ID, (b) the original Drenjongke sentence in Tibetan script, and (c) an English translation; and the token level, consisting

```
#sentence id = 112
#text = ཁྱན་ཀུས་ དབྱིན་ཇི་ ཀུསས་ཡོ་ གནང་ག་ ཡགས།
#trans = Do you speak English?
1   ཁྱན་ཀུས་     PRON        lhan gye    you (honorific)
2   དབྱིན་ཇི་     NOUN        y'in j'i    english
3   ཀུསས་ཡོ་     VERB        kyap po     doing
4   གནང་ག་      AUXHON+Q    nang ga     ?
5   ཡགས།        HON         lâ
```

Figure 1: Excerpt from the Drenjongke corpus

of (a) token, (b) PoS tag, (c) romanization of the token in phonological Drenjongke (Section 4.2), and (d) an English gloss of each token. An excerpt from the corpus is shown in Figure 1. The entire corpus is publicly available.[9]

We built the corpus using the following steps: (a) correcting errors in the text obtained by running Optical Character Reading (OCR) of scanned pages, (b) tokenizing sentences using spacing, and (c) manually annotating PoS tags of each token.

3.1 OCR and correcting OCR errors

We used Google OCR to recognize the Tibetan texts from scanned pages. The input method of the Tibetan script is complex because typing a syllable requires typing a root character that has additional superscript, subscript, and vowel diacritics. Since the size of our corpus is not large, we could have typed all the data, but we opted for using the OCR method instead. Testing the OCR method was beneficial because we found that the OCR-ed texts contained errors due to the "tsha-lag" ◌ྫ marker, which is used to mark the pronunciation of [b^{j}] in Drenjongke. The use of this marker is unique to Drenjongke because Tibetan does not have the sound [b^{j}]. Any errors with ◌ྫ were manually corrected. The knowledge of this shortfall of the Tibetan OCR for our corpus will help us when we add more text to the current corpus.

3.2 Tokenization

For tokenization, space was set as a delimiter. Drenjongke script is marked by a syllable marker called "tsheg" ་, and has a space between potential morpheme or word boundaries. The use of space in the orthography is specific to Drenjongke as other Tibetan languages do not utilize spacing in a sentence. We assume that either a morpheme boundary or a word boundary is demarcated by a space. The tokenization process revealed some inconsistencies in the use of spacing. Thus, it is not im-

[5] https://universaldependencies.org/u/pos/

[6] https://github.com/tibetan-nlp

[7] https://www.soas.ac.uk/cia/tibetanstudies/tibetan-in-digital-communications/

[8] https://zenodo.org/record/574878#.XvBraGozZZ1

[9] https://github.com/ICULingLab/drenjongke

mediately obvious whether spacing directly represents morpheme or word boundaries.

In addition to the spacing, defining words, affixes, and clitics in Drenjongke is not an easy task because monosyllabic morphemes can readily concatenate to the head of a word. For example, one verb + one monosyllabic affix can be counted as a word, while they can be two different words or a word and a clitic depending on the definition of the word. As such, spacing was used in the tokenization of the sentences from the operational point of view. In our annotation scheme, tokens with multiple PoS tags are joined with the "+" symbol: for example, VERB+INF (an infinitive marker).

3.3 Annotation

The annotation of the current corpus was done by the authors; the first two authors have worked on the language for a number of years, and the last author is a native speaker. Due to the lack of solid annotation guideline, we failed to locate other native speakers who can annotate the corpus. As such, no independent annotation verification could be performed.

The PoS tag design is based on the gloss used in Yliniemi (2019). Though applying the Universal PoS tagset can be one solution, using gloss-based tags makes the corpus more informative, especially because Drenjongke has multiple functional particles which would be reduced to PART (particles) or ADP (adpositions) in the Universal PoS tags. Though the use of the Universal features would be suitable, not all the features are covered in the list.[10] Thus, we are planning to develop a language-specific documentation of those two categories when applying the Universal Dependencies framework in the near future.

In this section, we list the PoS tags proposed for the Drenjongke corpus in 3.3.1. Section 3.3.2 lists meta information on the tokens and PoS tags. Section 3.3.3 presents three examples of multi-functional morphemes and investigates disambiguation criteria for each morpheme. We also discuss some difficulties in the PoS annotation of the Drenjongke language in 3.3.4 and in 3.3.5.

3.3.1 Part-of-Speech tag set

This section lists the 34 PoS tags proposed and used in the Drenjongke Corpus. Both 8 freestanding closed class PoS tags, 18 modifying closed class PoS tags and 8 open class PoS tags were used in the tagging process. The closed class tags mark relatively fixed vocabulary, while the open class unit can be labelled on newly created words. The modifiers PoS tags are always attached to other segments such as VERB or NOUN and can not appear on their own. Freestanding PoS tags do not have such restriction.

Closed Class (freestanding)

AUX Copulas.

AUXHON AUXHON stands for honorific auxiliary. Drenjongke has a complex honorific system, and the verb *give* is frequently used to express the honorific meaning. In this case, *give* is a secondary verb labelled as AUXHON, which will be explained in detail in 3.3.3.

AUXQ Copulas used in the interrogatives.

CCONJ Coordinating conjunction, དང་ 'd'ang' (and). Moreover, Drenjongke employs a different form as a verbal coordination conjunction སྟེ་ 'te'.

HON The honorific particle ལགས་ 'hnang/nang/hnâ' is often found at the end of a sentence or phrase, as well as attached to a person's name, indicating that the sentence is honorific.

NEGAUX Drenjongke has negative auxiliaries that are not easily decomposed into simply a negation part and a copular part.

NUM Numerals written with both digits and with characters. No ordinal numbers appear in the corpus.

PRON Pronouns, which can be followed by case markers, and in that case, they are marked PRON+POSS, PRON+DAT, PRON+LOC.

Closed Class (modifiers)

DAT Dative case marker.

DET Definite and indefinite articles: འདི་ 'di' (this) and ཁྱེ་ལས་ 'khye le' (all).

IMP A marker of imperative as a verbal suffix.

INF A marker of infinitive as a verbal suffix.

IPFV A marker of imperfective as a verbal suffix.

LOC Locative case marker.

MOD Modality, ཆུ་གས་ 'chu' (can) and དགོས་ 'gö' (need to). In most cases, these are attached to the verb; however, they can also follow a noun. In that case, the modality behaves as a normal verb "want, need" instead of "want to, need to." For example, ཆུ་ དགོས་པོ་ 'chu gö bo' (water need-INF).

[10] https://universaldependencies.org/u/feat/index.html

NEG Affixes that attach verbs and adjectives to render their meaning negative.

NMLZ A nominalizer that attaches to an auxiliary verb or a verb. English equivalent *-er*.

NPST A marker of non-past tense as a verbal suffix.

POSS Possessive case marker.

PREP Prepositions.

Q Question particles.

REFL Reflexives such as *own* preceded by a pronoun.

REL A relativizer meaning *where it does V*.

SCONJ Subordinate conjunction, ན་ 'na' (if), which follows verbs. ན་ itself is used in many different cases, and then needs to be distinguished when annotating (Section 3.3.3).

SIM A marker of simultaneity. It can be an adverbial element by itself or a post-posed element of the tensed verb form.

SUG Suggestive particle as a verbal suffix. It is equivalent to *let's* in English.

Open Class

ADJ Adjectives. Some adjectives share the morpheme -རྒས་.

ADV Adverbs.

INTJ Interjections, such as *Yes*, *No*, and *Hello*.

MWE Multiword expressions. One example of MWE in the corpus is བཀྲ་ཤིས་ བདེ་ལེགས། 'tra shi de lek' (Congratulations). བཀྲ་ཤིས་ means auspicious and བདེ་ལེགས་ means goodness, happiness.

NOUN Nouns. As with pronouns, nouns are also followed by case markers. The corresponding annotation is NOUN+POSS, NOUN+DAT, NOUN+LOC.

PROPN Proper nouns, place, and person names. Possibly followed by case markers as nouns.

UNK Unknowns. Annotation is made by consulting the English translation of each token, as well as looking up examples from the descriptive Drenjongke grammar (Yliniemi, 2019) glossed in similar categories with our PoSs. Nevertheless, it is sometimes not clear which PoS corresponds to a token or a morpheme since the word never appears in that grammar book. In those cases, we annotate them UNK.

VERB Verbs in Drenjongke are often followed by morphemes that indicate TAME (tense, aspect, mood, evidentiality).

3.3.2 Meta Information on PoS tag Structure

Our corpus contains 275 sentences and 1379 tokens. The small number of tokens in each sentence is due to 1) the type of genre of the Drenjongke text, and 2) the agglutinative morphology of Drenjongke. The original text comes from a book for a Drenjongke person so that they can learn basic phrases in Drenjongke. Sentences are conversational and thus are short in general. In particular, a large number of sentences are presented as pairs of questions and responses. The responses such as "Yes, we can" are shorter than full sentences.

Another factor that makes fewer tokens per sentence is the synthetic characteristics of Drenjongke morphology. It uses various verbal and nominal affixes to express tense, aspect, mood, evidentiality, etc. In the PoS tags employed in our corpus, these functional elements are analyzed as a unit, which further lessened the number of unique tokens.

Figure 2 shows the top-10 PoS-tagged sequence patterns observed in a token separated by spaces. A token is marked with more than one PoS tag. The high appearance frequency of the HON tag is one of the characteristics of Drenjongke. The HON tag often appears as one token or a part of a token preceded by an auxiliary verb.

3.3.3 Disambiguation of functional morphemes

Drenjongke has a variety of monosyllabic morphemes that have more than one function. In the annotation scheme, these morphemes need to be disambiguated, as in the three examples below.

ན་ 'na' (LOC vs. SCONJ)

ན་ is a morpheme that attaches to nouns, verb,s or adjective wh-words (such as *which*). When attached to a noun, ན་ functions as a locative morpheme (annotated as LOC). After a verb, it functions as a subordinating conjunction (annotated as SCONJ).

- ང་ སྒང་རྟོག་ན་ སྡོད་ཀྱི་ ཨིན་ལགས།ངོ་ གནང་ལགས།
  nga gangtok na dö to ing la
  "I live in Gangtok. (LOC)"

- ཁཱ་ངའང་ ཡོད་ན་ ངའི་རྩང་ ཇོན།
  ka ngal yod na ngê tsä j'on
  "If you have any difficulty then come to me. (SCONJ)"

ཨོ་ 'she' (NPST vs. NOUN)

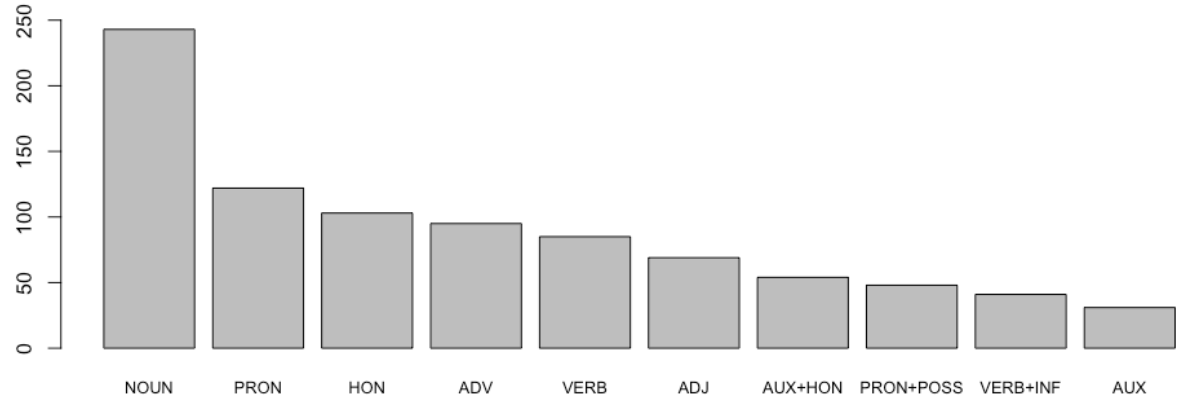


Figure 2: Top-10 PoS-tagged Sequence Patterns

ཤད་ only attaches to a verb. As a clause suffix, it indicates non-past tense (annotated as NPST). ཤད་ as a derivational suffix marks a gerund (annotated as NOUN reflecting the result of becoming a gerund).

- གཟིམ་པ་ བྱོན་ཤད་ ལ་གས།
  zim pa j'on she la
  "Going to sleep? (NPST)"
- འདི་ཁར་ གནས་ རྩ་ཆེན་ མཇལ་ཤད་ གན་ ཡོད་ལ་གས།
  di khâ hne tsä ch'e j'ä s'he gan yô la
  "Are there any important holy places around here? (NOUN)"

གནང་ 'hnang' (VERB vs. AUXHON)

གནང་ is employed as a main verb as well as a secondary verb. When used as a main verb, གནང་ is annotated as VERB. The usage as a secondary verb is annotated as AUXHON because it expresses an honorific register.

- ལྷན་རྒྱས་ ད་ལྟ་ གན་གནང་ བཟོ་ལ་གས།
  lhan gye däng ga na j'on bö lâ
  "What are you doing right now? (VERB)"
- དེ་རིང་རང་ གཡོག་འདི་ རྫོངས་པོ་ གནང་།།
  d'a ring ra yö di nyô po nang
  "Finish this work by today. (AUXHON)"

3.3.4 Annotation challenges

In our corpus, spacing in the written text was set as a delimiter. When a token has more than two morphemes, the PoS tags are concatenated by "+." This "+" method is often used in the Dzongkha corpus (Chungku et al., 2010) and Korean corpora[11], two languages with morphological structures akin to Drenjongke. When automatizing the process of PoS tag annotation, the presence of the "+" operand is non-trivial and any corpora building process must take that into consideration because the "+" operation is highly productive in Drenjongke.

3.3.5 Difficulty of evaluation

Language resource papers are often presented with metrics such as inter-annotator agreement to prove their qualities. However, it is difficult to obtain those metrics for low-resource languages that have a limited number of researchers. Moreover, difficulty lies in the fact that there is no ground truth for PoS tagging.

4 Discussion

As far as we know, no Drenjongke corpus has been available until now. Building a Drenjongke corpus was not a simple task and various considerations were discussed in the process. We present four points: (a) domain specificity of the corpus, (b) social impact of machine-readable resources, (c) language revitalization, and (d) annotation challenges.

4.1 Domain specificity

The Drenjongke corpus is based on a conversational textbook for Drenjongke beginners, targeting young Drenjongke people who do not always speak Drenjongke in their everyday life. A problem with this specific purpose of the text has pros

[11] see the fine-grained tag section of the following corpus for an example: https://github.com/UniversalDependencies/UD_Korean-Kaist/

in that the corpus has sentences relevant to average Drenjongke speakers. Cons of this specific text is the lack of diversity of genres in the data set. In order to address this issue, we are in the process of augmenting the data with texts from other genres such as folk songs and philosophical texts. The soon-to-be-expanded corpora will have annotations using syntactic relation tags adapted from the Universal Dependencies (UD) scheme, which has gained recognition as an optimal tool for a cross-lingual annotation scheme. Tibeto-Burman languages are still absent in the UD version 2.6 (Zeman et al., 2020).

4.2 Phonological Drenjongke

Drenjongke written in the Tibetan script follows the spelling norm of Classical Tibetan resulting in the inclusion of characters that are not pronounced. Written forms following Classical Tibetan are helpful for tracking etymological information, but they add difficulties in reading and writing, especially if one is not familiar with the written Tibetan forms.

The romanization of Tibetan scripts is based on the "Wylie system," which is a grapheme-based transliteration (Wylie, 1959). Among the community members, there is an increasing call for a romanization system that reflects the actual Drenjongke pronunciation. A recent invention of such a system with collaborative work with the Drenjongke community is the "Phonological Drenjongke" proposed in van Driem (2016). The romanization system in our corpus follows "Phonological Drenjongke" added by a native speaker researcher. A larger corpus in the future is expected to aid the development of an automatic transliteration system between Tibetan script and phonological Drenjongke.

4.3 Language documentation and revitalization

Building a Drenjongke corpus may serve as an effective way of documenting the language. One of the greatest challenges in creating this corpus was the lack of detailed morphological information on Drenjongke. The whole process needed constant communication among the three authors, one of whom is a native speaker of Drenjongke. This collaboration is essential in creating corpora that reflect the current Drenjongke language.

Spoken corpora of multiple Tibeto-Burman languages exist, but text-based corpora of endangered Tibeto-Burman languages are rare. The process of building this Drenjongke corpus emphasized the need for an active collaboration between native speaker researchers and NLP researchers, a challenge that is not easily addressable. The current team benefited from a linguistic project by the second author, which may be a way to move forward when interdisciplinary collaboration is needed.

5 Conclusion and Future work

This paper reports the building of a Drenjongke PoS-tagged corpus containing 275 sentences with English translation and romanization in "Phonological Drenjongke." Our future work lies in establishing a reliable word segmentation method as well as in optimizing the PoS tagging process while referring to the existing tools for Tibeto-Burman languages. Additionally, the development of the language-specific documentation for the Universal PoS tagset, as well as dependency relation labelling, remains for future work.

Acknowledgement

The authors are grateful to Jin-Dong Kim and three anonymous reviewers for their feedback on the paper, Mamoru Komachi for insightful discussions regarding the annotation process, Arseny Tolmachev for the post-acceptance mentorship, and the AACL-IJCNLP SRW committee members for providing support of various kinds. Of course, all remaining errors are of our own. Thanks also go to Jigmee Wangchuk Bhutia and Lopen Karma Gyaltsen Drenjongpo for allowing us to edit and publish the contents of the phrase book.

References

Chungku Chungku, Jurmey Rabgay, and Gertrud Faaß. 2010. Building NLP resources for Dzongkha: a tagset and a tagged corpus. In *Proceedings of the Eighth Workshop on Asian Language Resources*, pages 103–110.

Chenchen Ding, Hnin Thu Zar Aye, Win Pa Pa, Khin Thandar Nwet, Khin Mar Soe, Masao Utiyama, and Eiichiro Sumita. 2019. Towards Burmese (Myanmar) Morphological Analysis: Syllable-Based Tokenization and Part-of-Speech Tagging. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(5):1–34.

Chenchen Ding, Sann Su Su Yee, Win Pa Pa, Khin Mar Soe, Masao Utiyama, and Eiichiro Sumita. 2020. A Burmese (Myanmar) Treebank: Guideline and Analysis. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19(40):1–13.

Lopen Karma Gyaltsen Drenjongpo. 2017. *Bhutia learning book for young beginners [’lo sar zhön bo tsu lo phen be lho ke jöng d’eb]*. Phen-De Ga-Tshal Ling Dharma Centre, Sikkim, India.

George van Driem. 2016. The phonology of Dränjoke: Experimental development of Roman Dränjoke and Phonological Dränjoke. *Manuscript. University of Bern*.

Nathan W Hill and Johann-Mattis List. 2017. Challenges of annotation and analysis in computer-assisted language comparison: A case study on Burmish languages. In *Yearbook of the Poznan Linguistic Meeting*, volume 3.1, pages 47–76. Sciendo.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Huidan Liu and Long Congjun. 2018. CTTC: A Collection of Tibetan Text Corpora . In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 13–19.

Sithar Norbu, Pema Choejey, Tenzin Dendup, Sarmad Hussain, and Ahmed Muaz. 2010. Dzongkha word segmentation. In *Proceedings of the Eighth Workshop on Asian Language Resouces*, pages 95–102.

Turrell Wylie. 1959. A standard system of tibetan transcription. *Harvard journal of Asiatic studies*, 22:261–267.

Juha Yliniemi. 2019. *A descriptive grammar of Denjongke (Sikkimese Bhutia)*. Ph.D. thesis, University of Helsinki.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, et al. 2020. Universal Dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Towards a Standardized Dataset on Indonesian Named Entity Recognition

Siti Oryza Khairunnisa **Aizhan Imankulova** **Mamoru Komachi**
Tokyo Metropolitan University
6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
{siti-oryza-khairunnisa, imankulova-aizhan}@ed.tmu.ac.jp
komachi@tmu.ac.jp

Abstract

In recent years, named entity recognition (NER) tasks in the Indonesian language have undergone extensive development. There are only a few corpora for Indonesian NER; hence, recent Indonesian NER studies have used diverse datasets. Although an open dataset is available, it includes only approximately 2,000 sentences and contains inconsistent annotations, thereby preventing accurate training of NER models without reliance on pre-trained models.

Therefore, we re-annotated the dataset and compared the two annotations' performance using the Bidirectional Long Short-Term Memory and Conditional Random Field (BiLSTM-CRF) approach. Fixing the annotation yielded a more consistent result for the organization tag and improved the prediction score by a large margin. Moreover, to take full advantage of pre-trained models, we compared different feature embeddings to determine their impact on the NER task for the Indonesian language.

1 Introduction

Named entity recognition (NER) is an essential subtask in natural language processing (NLP). However, NER still suffers from data sparseness for the majority of languages, including Indonesian.

Various Indonesian NER approaches have been proposed, ranging from rule-based methods (Budi et al., 2005) to machine learning-based techniques (Leonandya et al., 2015; Aryoyudanta et al., 2016). The DBpedia and Wikipedia datasets are mainly used for supervised approaches (Alfina et al., 2016; Leonandya et al., 2015; Aryoyudanta et al., 2016; Gunawan et al., 2018). Other datasets include Twitter (Taufik et al., 2016; Wintaka et al., 2019) and conversational datasets from chatbots (Kurniawan and Louvan, 2018), but the sizes of these datasets are limited. Unfortunately, almost all of the previous studies on Indonesian NER did not release their datasets, which provide essential information for machine learning-based NLP.

One Indonesian NER dataset with human annotation that is openly available is a news dataset from Syaifudin and Nurwidyantoro (2016) (hereinafter referred to as S&N (2016)). However, this dataset exhibits inconsistency problems. In this study, we re-annotated this dataset, thereby developing a more standardized Indonesian NER resource to improve the NLP foundation for the Indonesian language. The most problematic entity is organization, followed by location and person. Certain tokens had been tagged as entities that they were not; for example, the term "DPP" (which means "party's representative council") is not an organization name but had been tagged as such.

The most recent Indonesian NER work that used BiLSTM-CRF was conducted by Wintaka et al. (2019) using FastText as the word representation. It has been claimed that FastText offers advantages in handling misspelled words and out-of-vocabulary (OOV) problems (Bojanowski et al., 2017). Therefore, we used BiLSTM-CRF with FastText as our baseline model.

We also experimented with Bidirectional Encoder Representations from Transformers (BERT), a transformer-based language model known to work best in various tasks in NLP as well as NER by acquiring contextual word meanings based on their usage in a sentence (Devlin et al., 2019). For the experiment, we compared three models: the multilingual transformer-based models; mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) and a monolingual BERT for the Indonesian language (IndoBERT) (Wilie et al., 2020). Regarding the limited vocabulary in our low-resourced data, we hypothesized that these embeddings could solve the OOV problem because of the vocabu-

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 64–71

lary coverage in the large-scale data used for pre-training those embeddings.

Our contributions can be summarized as follows:

1. We re-annotated the human-annotated Indonesian NER dataset to improve its consistency and made the dataset publicly available.[1]
2. We analyzed the impact of the data consistency by comparing the performance of NER models trained on the previous and re-annotated datasets. Our dataset significantly improved NER performance.
3. We compared the static and dynamic word embeddings for the Indonesian NER task and showed the impact of different embeddings on the NER model performance.

2 Related Works

Several recent NER methods employed the bidirectional neural network and a conditional random field (CRF) as the encoder–decoder layer (Lample et al., 2016; Peters et al., 2018; Akbik et al., 2018). Using contextual information as a representation input to the encoder model improved the score substantially as it helped the model learn the context of the entities. Akbik et al. (2018) introduced Flair embedding, a contextual string embedding approach, and showed that stacking word and character embeddings increased the model's ability to understand contextual and word-level semantic representations. The use of a Transformer (such as in BERT) also demonstrated significant results for numerous NLP downstream tasks, for example, by BERT, which was proven to work on certain tasks, as well as for NER (Devlin et al., 2019; Conneau et al., 2020). The current state-of-the-art NER model to date was created by fine-tuning a cloze-driven pre-trained bidirectional transformer model (Baevski et al., 2019).

Nevertheless, we focus on low-resourced language NER. The majority of previous studies on low-resourced NER also implemented BiLSTM-CRF as the sequence labeling method and experimented with input representation (Pham and Le-Hong, 2018; Poostchi et al., 2018; Singh et al., 2019). BERT has also been employed in several low-resource languages, including Bulgarian (Marinova, 2019), Arabic (Antoun et al., 2020), and Basque (Agerri et al., 2020).

Deep learning has recently been used in Indonesian NER research. The most widely used method is the BiLSTM algorithm (Huang et al., 2015; Lample et al., 2016). Various input representation methods have been applied, such as convolutional neural networks (CNNs) for word n-gram representation (Gunawan et al., 2018) and pre-trained word embeddings with part-of-speech (PoS) tags (Hoesen and Purwarianti, 2018). In exploring the OOV problem in conversational text, Kurniawan and Louvan (2018) also employed BiLSTM-CRF without including any pre-trained word representation. In recent work by Wintaka et al. (2019), the same neural sequence labeling model was implemented, and pre-trained FastText Indonesian word embedding was applied as the input. In work similar to ours, Leonandya and Ikhwantri (2019) investigated the impact of language model pre-training on the NER task. However, their conversational texts data is not publicly available, and therefore their study is not replicable. The latest Indonesian NER work is from Wilie et al. (2020), who built and fine-tuned an Indonesian BERT (IndoBERT) pre-trained model for twelve NLP tasks, including NER.

Previous work using the same dataset was conducted by S&N (2016) for a quotation identification task. The dataset was constructed from three Indonesian online news sites, namely Kompas[2], Tempo[3], and TribunNews[4]. The topics covered by this dataset mainly concern politics, society, and economics. In this task, the data were labeled for quotation identification. However, the NER data were manually tagged as well, as they were used in preprocessing for the quotation identification task. In this study, we focused on the NER task and re-annotated the data because of the inconsistency.

3 Methodology

3.1 Inconsistency of Existing Dataset

We used an open dataset released by S&N (2016), which is available on GitHub[5]. However, we found that several tokens in the dataset were not tagged correctly. For example, tokens of certain organizations and persons were not tagged or were tagged incorrectly. Table 1 shows an examples of inconsistency in the annotation. The three sentences

[1] https://github.com/khairunnisaor/idner-news-2k

[2] https://www.kompas.com/

[3] https://www.tempo.co/

[4] https://www.tribunnews.com/

[5] https://github.com/yusufsyaifudin/Indonesia-ner

Sentence 1	President Joko Widodo met the chairman of Gerindra Prabowo Subianto at Istana Bogor											
Indonesian	**Presiden**	**Joko**	**Widodo**	**bertemu**	**Ketua**	**Umum**	**Gerindra**	**Prabowo**	**Subianto**	**di**	**Istana**	**Bogor**
English translation	President	Joko	Widodo	met	chairman	general	Gerindra	Prabowo	Subianto	at	palace	Bogor
S&N (2016)	O	B-PER	I-PER	O	B-PER	I-PER	I-PER	I-PER	I-PER	O	B-LOC	I-LOC
Ours	O	B-PER	I-PER	O	O	O	B-ORG	B-PER	I-PER	O	B-LOC	I-LOC
Sentence 2	Politician PDI Perjuangan Guruh Soekarnoputra visited the chairman of Gerindra Party Suhardi											
Indonesian	**Politikus**	**PDI**	**Perjuangan**	**Guruh**	**Soekarnoputra**	**menjenguk**	**Ketua**	**Umum**	**Partai**	**Gerindra**	**Suhardi**	
English translation	Politician	PDI	Perjuangan	Guruh	Soekarnoputra	visited	chairman	general	Party	Gerindra	Suhardi	
S&N (2016)	O	B-ORG	I-ORG	B-PER	I-PER	O	O	O	O	O	B-PER	
Ours	O	B-ORG	I-ORG	B-PER	I-PER	O	O	O	B-ORG	I-ORG	B-PER	
Sentence 3	Vice chairman of Gerindra Party Edy Prabowo stated that his party would not be hurt											
Indonesian	**Wakil**	**Ketua**	**Umum**	**Partai**	**Gerindra**	**Edy**	**Prabowo**	**menyatakan**	**partainya**	**tak**	**akan**	**sakit hati**
English translation	Vice	chairman	general	Party	Gerindra	Edy	Prabowo	stated	his party	not	would	hurt
S&N (2016)	O	O	O	B-ORG	I-ORG	B-PER	I-PER	O	O	O	O	O
Ours	O	O	O	B-ORG	I-ORG	B-PER	I-PER	O	O	O	O	O

Table 1: Examples of tags before (S&N (2016)) and after (ours) re-tagging. The red tokens indicate the difference after re-tagging. The blue tokens represent consistent annotation between S&N (2016) and ours. Tag prefix meanings: B indicates the entity's first word, whereas I indicates the second and remaining part of the entity.

Data Split	**Sentences**	**Tokens**
Train	1,464	30,248
Development	367	7,863
Test	509	10,588
Total	2,340	48,699

Table 2: Data statistics.

		Our Tags				# of tags
		LOC	ORG	PER	O	
S&N Tags	LOC	1,153	26	4	344	1,527
	ORG	5	1,562	2	78	1,647
	PER	4	3	2,317	39	2,363
	O	91	701	127	42,241	43,160
# of tags		1,253	2,292	2,450	42,702	

Table 3: Confusion matrix of our re-annotation from S&N (2016). The number of tags is represented at the token level. The first column indicates the entity's previous tag and the header denotes our tag in the re-annotation. LOC: location; ORG: organization; PER: person; O: other.

have a pattern of [title] [organization] [person]. In Sentence 1, S&N (2016) tagged all tokens *"Ketua Umum"* (title), *"Gerindra"* (organization name) and *"Prabowo Subianto"* (person name) as person names. In the red part of Sentence 2, S&N (2016) only tagged *"Suhardi"* (person name) as a person name and did not tag *"Partai Gerindra"* as an organization name. In the blue part of Sentences 2 and 3, S&N (2016) did not tag the titles (*"Politikus"* and *"Wakil Ketua Umum"*) as entities but did tag the organization and person name correctly. These sentences demonstrate that there were three different annotations for the same sentence pattern. As this phenomenon appeared several times, we examined the dataset and checked the annotations one by one.

3.2 Dataset Re-annotation

To address these problems, we asked three native speakers to re-annotate the data manually. Although S&N (2016) covers five entities, we only re-annotated only three entities that are commonly used in the NER task, namely location, organization, and person. We omitted the other two, time and quantity, because we wanted to build a model with a strong foundation for recognizing ambiguous nouns. As most time and quantity entities are written in numeric form, they are easy to be recognized by a well-developed NER model. In the experiment, we considered just the three entities in both datasets so that the results would be reasonably comparable.

We calculated the inter-annotator agreement of the three annotators and obtained an agreement score of 0.92 using Fleiss' kappa (Fleiss, 1971), which indicates a high agreement and good reliability (Artstein and Poesio, 2008). In this study, we used the same split as that used in S&N (2016). However, owing to the absence of the development set as in theirs, we randomly sampled data from the training set to constitute the development set, as indicated in Table 2.

3.3 Our Annotation Guidelines

To clearly differentiate how we annotated each entity, here we provide the guidelines we used in re-annotating the dataset.

- `Location`: indicates the name of a location name where activities or events happened semantically. Such an entity is usually preceded by a location preposition, namely *"di"* (at),

"ke" (to), or *"dari"* (from). Specific location names such as a country or city name (e.g., Indonesia in "Indonesia is one of the largest countries") when not used contextually as a location would not be annotated as a location. An organization name (e.g., university or office), conversely, is sometimes used as a location name when the sentence refers to its building or location. In this case, we annotate the entity as a location name.

- `Organization`: indicates an organization's name. The name of the organization is usually an official institution that is legally registered.

- `Person`: identifies a person's name. Any form of the person's name—full, nickname, or abbreviation—is annotated as one name. For example, *"Abu Rizal Bakrie"* is the full name of a person, who may also be mentioned as *"Ical"* (nickname) or *"ABR"* (abbreviation). A person's title, such as *"Pak"* (Mr.) in *"Pak Ical"* (Mr. Ical) is not included in the person's name; it is annotated as *"Pak [Ical]$_{B\text{-}PER}$"*, not as *"[Pak]$_{B\text{-}PER}$ [Ical]$_{I\text{-}PER}$"*.

- An organization or person name that is sometimes written in full may, at other times, be written in its abbreviated form. When both forms appear, the annotation will be separated into two entities. For example, the sentence, *"Universitas Gadjah Mada (UGM) berlokasi di Yogyakarta."* (Gadjah Mada University (UGM) is located in Yogyakarta) is annotated, as shown below:

  [Universitas]$_{B\text{-}ORG}$ [Gadjah]$_{I\text{-}ORG}$ [Mada]$_{I\text{-}ORG}$ ([UGM]$_{B\text{-}ORG}$) berlokasi di [Yogyakarta]$_{B\text{-}LOC}$

For ambiguous entities, the tags were determined according to the word's contextual use; that is, whether it is used as a location name or organization name (these two entities are generally the most confusing). When there was a disagreement, the tag chosen by the majority of the taggers was determined as the final tag.

Table 3 presents the confusion matrix of our re-annotation of S&N (2016). The number of tags for the organization and person entities increased after the re-tagging. Meanwhile, the number of location entities was reduced from approximately 1,500 to 1,200, and the other (O) entity was reduced by almost 500 tags. This indicates that 20% of the location tags were incorrect, and almost 500 tokens were not tagged. Comparing our annotation, we calculated the percent difference by dividing the difference in the number of tags by the total number of tokens.

4 Experiment

4.1 NER Methods

BiLSTM-CRF is a deep learning algorithm introduced by Huang et al. (2015) and has mainly been used for the NER task owing to its ability to solve sequence tagging problems. Following its successes in dealing with English NER tasks (Lample et al., 2016; Akbik et al., 2018; Ma and Hovy, 2016), BiLSTM-CRF was also implemented in recent Indonesian NER; relatively good results were obtained compared to those of rule-based and former machine learning approaches (Hoesen and Purwarianti, 2018; Kurniawan and Louvan, 2018; Wintaka et al., 2019). We employed a method used by Wintaka et al. (2019) as our baseline. They used FastText (Bojanowski et al., 2017), a pre-trained word embedding with sub-word features, as the input representation for the BiLSTM-CRF.

In addition to FastText, we also used some pre-trained multilingual and monolingual models as the input representation for BiLSTM-CRF. For multilingual models, we applied multilingual BERT (mBERT) by Devlin et al. (2019) and XLM-R by Conneau et al. (2020), and for the monolingual model, we applied the Indonesian monolingual BERT pre-trained model IndoBERT by Wilie et al. (2020). To compare the use of BERT with the feature representation approach, we also investigated the potential benefit of using a fine-tuning approach with all BERT pre-trained models.

4.2 Settings

We used the implementation of the BiLSTM-CRF approach provided by Flair NLP[6], a simple framework for sequence labeling tasks (Akbik et al., 2018). We identified three entity types, namely location (LOC), organization (ORG), and person (PER), and used the IOB format defined by Tjong Kim Sang (2000). We conducted five experiments for each model and calculated the average scores. We applied two approaches for the NER task implementation.

First, in the feature-based approach, we used BiLSTM-CRF and experimented with several word

[6] https://github.com/flairNLP/flair

Dataset Annotation		Overall Scores			LOC			ORG			PER		
Train	**Test**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
S&N	S&N	69.20	84.55	76.11	74.82	85.78	79.92	45.54	83.02	58.82	83.60	84.86	84.23
Ours	S&N	66.39	88.28	75.79	79.05	85.57	82.18	39.64	88.95	54.84	84.95	88.60	86.74
S&N	Ours	89.27	80.06	84.41	80.70	85.19	82.88	88.12	71.21	78.77	92.13	85.91	88.91
Ours	Ours	**92.23**	**89.52**	**90.85**	**89.02**	**88.52**	**88.76**	**89.13**	**88.13**	**88.63**	**95.50**	**90.83**	**93.10**

Table 4: Baseline model comparison of S&N (2016)'s and our annotation performance. The bold scores show the best score for both models when tested on our test set, and the underlined scores present the best score when tested on S&N (2016)'s test set.

Models	**Features**	Overall Scores			F_1 Scores for Each Tag		
		P	**R**	**F**	**LOC**	**ORG**	**PER**
BiLSTM-CRF	FastText (baseline)	92.23	89.52	90.85±0.08	88.76	88.63	93.10
	mBERT	90.11	89.80	89.95±0.05	87.30	86.54	93.20
	XLM-R	91.12	94.05	92.56±0.54	89.01	88.95	96.25
	IndoBERT	**94.93**	**94.87**	**94.90±0.31**	**89.84**	**92.99**	**97.49**
Fine-tune mBERT	–	89.54	92.33	90.91±0.51	**86.73**	87.78	94.11
Fine-tune XLM-R	–	91.60	**94.76**	**93.15±0.23**	85.59	90.46	**97.41**
Fine-tune IndoBERT	–	**91.66**	94.64	93.13±0.31	83.84	**91.81**	96.83

Table 5: BiLSTM-CRF model performance for contextual embedding experiment.

embeddings as the input representation, namely FastText (Grave et al., 2018), mBERT (Devlin et al., 2019) [7], XLM-R (Conneau et al., 2020), and IndoBERT (Wilie et al., 2020); these are shown in Table 5. The parameter settings were as follows: a learning rate of 0.1, a dropout of 0.5, a mini-batch size of 32, a maximum of 200 epochs, one BiLSTM hidden layer, and 256 BiLSTM hidden units. The implementation used an early stopping method, where the model would stop training when the loss score did not improve in a sequence of five epochs.

Second, we fine-tuned all of the BERT models for the NER task. The output layer used for fine-tuning the BERT models was the standard softmax layer. For the fine-tuning settings, we set the batch size to 32, the number of epochs to five, and the learning rate to [3e-5, 5e-5, 7e-5] respectively.

5 Results

Annotation performance. Table 4 presents a comparison of the annotation performance between S&N (2016) and our annotation. We conducted a cross test for each model on both annotations' test sets with the aim of observing both models' performance and comparing them on an equal footing by testing them on the same annotations. For the S&N (2016), our baseline model provided an F_1 score of 76.11. Our annotation obtained a higher score, 84.41. Particularly in the organization tag, the score jumped relatively high by almost 20 points.

The phenomenon whereby many organization tokens were not tagged accounted for a sharp decrease in the F_1 score of the ORG tag in the S&N (2016) dataset (ORG: 54.84, LOC: 82.18 and PER: 86.74). Using our test data, we obtained a very high overall score with more consistent performance of the organization tag, as indicated by all three tags sharing relatively similar scores.

Multilingual vs. monolingual pre-trained models. Table 5 presents the results using pre-trained BERT models for our Indonesian NER task. The best result was obtained from the feature-based approach with the IndoBERT pre-trained model as the input representation for the BiLSTM-CRF architecture. Fine-tuning those models resulted in scores slightly under those of the feature-based approach. Regarding multilingual model performance, the XLM-R outperformed the mBERT model owing to its more extensive unsupervised multilingual data when pre-trained (Conneau et al., 2020).

The monolingual model performed better than the multilingual model when used as the feature representation for BiLSTM-CRF; the multilingual models are better when fine-tuned. Wilie et al.

[7] https://huggingface.co/transformers/pretrained_models.html

Sentence 1	Joko Widodo met Gerindra's Chairman Prabowo Subianto							
Indonesian	**Joko**	**Widodo**	**bertemu**	**Ketua**	**Umum**	**Gerindra**	**Prabowo**	**Subianto**
English translation	Joko	Widodo	met	chairman	general	Gerindra	Prabowo	Subianto
S&N (2016) annotation	B-PER	I-PER	O	B-PER	I-PER	I-PER	I-PER	I-PER
S&N (2016) FastText	B-PER	I-PER	O	O	O	O	B-PER	I-PER
Our annotation	B-PER	I-PER	O	O	O	B-ORG	B-PER	I-PER
Our FastText	B-PER	I-PER	O	O	O	B-ORG	B-PER	I-PER
Sentence 2	Required by the Ministry of Law and Human Rights							
Indonesian	**disyaratkan**	**oleh**	**Kementerian**	**Hukum**	**dan**	**Hak**	**Asasi**	**Manusia**
English translation	required	by	ministry	law	and	right	basic	human
S&N (2016) annotation	O	O	O	O	O	O	O	O
S&N (2016) FastText	O	O	B-ORG	I-ORG	I-ORG	I-ORG	I-ORG	I-ORG
Our annotation	O	O	B-ORG	I-ORG	I-ORG	I-ORG	I-ORG	I-ORG
Our FastText	O	O	B-ORG	I-ORG	I-ORG	I-ORG	I-ORG	I-ORG

Table 6: Examples of errors in prediction comparing S&N (2016) and our annotation when trained using the baseline BiLSTM-CRF model. Red indicates incorrect tokens and blue indicates the correct ones.

(2020) stated that the XLM-R model might achieve a better result on the NER task as entity names usually come from English or other languages. However, most of our dataset's entity names are in Indonesian, and hence, the IndoBERT pre-trained model supports this condition better. Additionally, mBERT and XLM-R use WordPiece (Wu et al., 2016) and SentencePiece (Kudo, 2018) tokenization respectively, whereby longer tokens are split into more common tokens. Multilingual models contain many languages in their vocabularies. When the pre-trained model is frozen for feature representation use, it is possible that some Indonesian sub-words are biased because they share representations with other language's sub-words. The IndoBERT was trained on the `Indo4B` dataset, which also contains Indonesian news corpora (Wilie et al., 2020). We hypothesize that the BiLSTM-CRF architecture fits better with our sequence classification task, supported by the rich Indonesian vocabularies covered by IndoBERT and the domain similarity between the `Indo4B` and our dataset. This is reflected by the very high organization scores obtained from IndoBERT models in both approaches.

6 Discussion

Table 6 presents some examples of errors by the model trained on S&N (2016) and on our annotation. In Sentence 1, the words *"Ketua Umum Gerindra"* were tagged as part of a person's name, although they are not. The S&N (2016) model identified *"Prabowo Subianto"* correctly as a person's name but did not tag *"Gerindra"* as an organization name. Meanwhile, our model correctly tagged *"Ketua Umum"* as not being any entity and *"Gerindra"* as an organization. In Sentence 2, the S&N (2016) annotation did not tag any of the words. However, the baseline model trained on S&N (2016) and on our annotation recognized the tokens as an organization name in all cases. These examples demonstrate that errors in an annotation could affect the prediction performance and decrease the model's score.

7 Conclusions and Future Work

We re-annotated the human-annotated Indonesian NER dataset to produce a more consistent annotation in the Indonesian NER task. This re-annotation obtained an F_1 score of 90.85 when using the baseline BiLSTM-CRF model was used with FastText. Our implementation also demonstrated that the use of pre-trained transformer-based language models, both the multilingual and the monolingual models, yielded better prediction results. Although the performance of Indonesian NER using either BiLSTM-CRF or fine-tuning depends on the pre-trained language model, we found that IndoBERT works best when using BiLSTM-CRF architecture, compared to the fine-tuning approach.

In the future, we plan to address word ambiguity in Indonesian by creating a gazetteer to add more supervision and perform distant supervised learning to aid the model in differentiating a word to be classified as each entity as in Nooralahzadeh et al. (2019). Also, we would like to work on other techniques such as transferring knowledge using a teacher-student learning from a high resource language such as English to a low-resource, such as Indonesian (Wu et al., 2020; Sun et al., 2019).

References

Rodrigo Agerri, Iñaki S. Vicente, Jon A. Campos, Ander Barrena, Xabier Saralegi, Aitor Soroa, and Eneko Agirre. 2020. Give your text representation models some love: the case for Basque. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4781–4788.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Ika Alfina, Ruli Manurung, and Mohamad I. Fanany. 2016. DBPedia entities expansion in automatically building dataset for Indonesian NER. In *Proceedings of the 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 335–340.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15.

Ron Artstein and Massimo Poesio. 2008. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Bayu Aryoyudanta, Teguh B. Adji, and Indriana Hidayah. 2016. Semi-supervised learning approach for Indonesian named entity recognition (NER) using co-training algorithm. In *Proceedings of the 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 7–12.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5360–5369.

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.

Indra Budi, Stéphane Bressan, Gatot Wahyudi, Zainal A. Hasibuan, and Bobby A. A. Nazief. 2005. Named entity recognition for the Indonesian language: Combining contextual, morphological and part-of-speech features into a knowledge engineering approach. In *Discovery Science*, pages 57–69.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378–382.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3483–3487.

William Gunawan, Derwin Suhartono, Fredy Purnomo, and Andrew Ongko. 2018. Named-entity recognition for Indonesian language using bidirectional LSTM-CNNs. In *Proceedings of the 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018): Empowering Smart Technology in Digital Era for a Better Life*, volume 135, pages 425 – 432.

Devin Hoesen and Ayu Purwarianti. 2018. Investigating Bi-LSTM and CRF with POS tag embedding for Indonesian named entity tagger. In *Proceedings of the 2018 International Conference on Asian Language Processing (IALP)*, pages 35–38.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv e-prints*, page arXiv:1508.01991.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Kemal Kurniawan and Samuel Louvan. 2018. Empirical evaluation of character-based model on neural named-entity recognition in Indonesian conversational texts. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 85–92.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Rezka A. Leonandya, Bayu Distiawan, and Nursidik H. Praptono. 2015. A semi-supervised algorithm for Indonesian named entity recognition. In *Proceedings of the 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pages 45–50.

Rezka A. Leonandya and Fariz Ikhwantri. 2019. Pre-trained language model transfer on neural named entity recognition in Indonesian conversational texts. In *Proceedings of the 33rd Pacific Asia Conference on Language, Information, and Computation*, pages 104–113.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Iva Marinova. 2019. Evaluation of stacked embeddings for Bulgarian on the downstream tasks POS and NERC. In *Proceedings of the Student Research Workshop Associated with Recent Advances in Natural Language Processing (RANLP) 2019*, pages 48–54.

Farhad Nooralahzadeh, Jan Tore Lønning, and Lilja Øvrelid. 2019. Reinforcement-based denoising of distantly supervised NER with partial annotation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 225–233.

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.

Thai-Hoang Pham and Phuong Le-Hong. 2018. End-to-end recurrent neural network models for Vietnamese named entity recognition: Word-level vs. character-level. In *Proceedings of the 2017 International Conference of the Pacific Association for Computational Linguistics*, pages 219–232.

Hanieh Poostchi, Ehsan Z. Borzeshi, and Massimo Piccardi. 2018. BiLSTM-CRF for Persian named-entity recognition ArmanPersoNERCorpus: the first entity-annotated Persian dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 4427–4431.

Oyesh M. Singh, Ankur Padia, and Anupam Joshi. 2019. Named entity recognition for Nepali language. In *Proceedings of the 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 184–190.

Linghao Sun, Huixiong Yi, Yong Liu, Huanhuan Chen, and Chunyan Miao. 2019. Back Attention Knowledge Transfer for Low-resource Named Entity Recognition. *arXiv e-prints*, page arXiv:1906.01183.

Yusuf Syaifudin and Arif Nurwidyantoro. 2016. Quotations identification from Indonesian online news using rule-based method. In *Proceedings of the 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 187–194.

Natanael Taufik, Alfan F. Wicaksono, and Mirna Adriani. 2016. Named entity recognition on Indonesian microblog messages. In *Proceedings of the 2016 International Conference on Asian Language Processing (IALP)*, pages 358–361.

Erik F. Tjong Kim Sang. 2000. Text chunking by system combination. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, pages 151–153.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, X. Li, Zhi Yuan Lim, S. Soleman, R. Mahendra, Pascale Fung, Syafri Bahar, and A. Purwarianti. 2020. IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Deni C. Wintaka, Moch A. Bijaksana, and Ibnu Asror. 2019. Named-entity recognition on Indonesian tweets using bidirectional LSTM-CRF. In *Proceedings of the 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019): Enabling Collaboration to Escalate Impact of Research Results for Society*, volume 157, pages 221 – 228.

Qianhui Wu, Zijia Lin, Börje Karlsson, Jian-Guang Lou, and Biqing Huang. 2020. Single-/multi-source cross-lingual NER via teacher-student learning on unlabeled data in target language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6505–6514.

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, Łukasz 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 e-prints*, page arXiv:1609.08144.

Resource Creation and Evaluation of Aspect Based Sentiment Analysis in Urdu

Sadaf Rani
Department of Computer Science,
Comsats University Islamabad,
Lahore Campus
1.5 km Defence Road, Off Raiwind Road,
Lahore, Punjab, Pakistan
sdf.rani0@gmail.com

Muhammad Waqas Anwar
Department of Computer Science,
Comsats University Islamabad,
Lahore Campus
1.5 km Defence Road, Off Raiwind Road,
Lahore, Punjab, Pakistan
waqasanwar@cuilahore.edu.pk

Abstract

Along with the rise of people generated content on social sites, sentiment analysis has gained more importance. Aspect Based Sentiment Analysis (ABSA) is a task of identifying the sentiment at aspect level. It has more importance than sentiment analysis from commercial point of view. To the best of our knowledge, there is very few work on ABSA in Urdu language. Recent work on ABSA has limitations. Only predefined aspects are identified in a specific domain. So our focus is on the creation and evaluation of dataset for ABSA in Urdu language which will support multiple aspects. This dataset will provide a baseline evaluation for ABSA systems.

1 Introduction

We are living in a world where web interaction is increasing. People share their emotions and express their feelings on different platforms through internet. A lot of work has been done to obtain valuable information from these reviews.

Sentiment analysis is a task of identifying people's emotions, feelings and opinion about a particular object from their reviews (Khan et al., 2018a). It is divided into three levels. Document level, Sentence level and Aspect level. "Document level" classifies the opinion of a complete document or paragraph into positive, negative, conflict or neutral. "Sentence level" classifies the sentiment of a sentence into positive, negative, conflict or neutral. "Entity/aspect level" It obtains the sentiment of a sentence at aspect (features) level (Patil and Yalagi, 2016) . Sentiment analysis has obtained importance in different areas like business intelligence and social media monitoring etc. Another area of interest is monitoring political data (Gold et al., 2018).

There has been very little research in the Urdu language since it is a low resourced language (Syed et al., 2010) Urdu has different morphological structures and linguistic features, so sentiment analysis intended for the English language can't be utilized for this language (Khan et al., 2018a) , (Syed et al., 2010). Many approaches and difficulties are discussed for the English language but very few for the Urdu language (El-Masri et al., 2017). Urdu is a mixture of many languages, so one of the challenging tasks is to handle different languages with their orientation (Khan et al., 2018b).

Aspect based sentiment analysis is a process of extracting the opinions expressed about a specific entity.

The sentiment analysis task becomes more challenging when there are multiple sentiments in a review on different aspects (Al-Smadi et al., 2015). Recently, the sentiment analysis research has moved towards having all the more fine-grained approaches thinking about the Urdu language aspects. But still, Urdu language is struggling at the aspect level. Recent work on the Urdu language at aspect level is not much more fine-grained as they deal only 4 predefined aspects. (ul Haq et al., 2020).

We aim to improve the current work by dealing the multiple aspects. An aspect can be a single word(e.g., Runs) or multi word (e.g., T 20 Series). A sentence may have one or more aspects that belongs to the same category. For this purpose clustering is used to group the related aspects into a single cluster. In this way we have a more fine-grained work at aspect level. We present a corpus of "Cricket" and "Football" domain in the Urdu language for ABSA. Which will provide a baseline evaluation and can further used for measuring the results of ABSA systems.

The rest of the paper is sorted out as follows. Section 2 describes related work. Section 3 contains a problem statement. Section 4 has

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 72–77

a proposed methodology. Section 5 contains a description of the data collection and annotation procedure. Section 6 provides information on guidelines for annotation. Section 7 has a conclusion.

2 Related work

The task of ABSA was first presented by SemEval in 2014 for English language (Pontiki et al., 2014). They provided data set of restaurant and laptop reviews for training and testing. (Al-Smadi et al., 2015) a benchmark dataset of Arabic language for aspect based sentiment analysis named as "HAAD" is prepared by them. They manually annotated the data at sentence level. Annotators used an online tool (BRAT) for annotation. This dataset provides the baseline evaluation to the four tasks i.e.; Aspect Extraction, Aspect Polarity, Aspect Category identification, Aspect Category Polarity.

(Rehman and Bajwa, 2016) a lexicon based approach has been adapted for sentiment analysis in Urdu language. They collected 124 comments from news sites. In preprocessing, tokens were generated. Polarity of each word is calculated by comparing with the sentiment lexicon. Accuracy of their approach was round about 66%.

(Arif et al., 2016) performed analysis on roman Urdu and provided the results by applying different classifiers on the dataset. Machine learning algorithms are applied on selected features for binary classification. Multiple classifiers are used on sparse matrix to evaluate the performance. Tf-idf is the term weighting model which gives the best overall accuracy with machine learning classifiers. SVM performs better then all of the classifiers by giving 96% accuracy.

Two datasets of low resourced languages Catalan and Basque for aspect level sentiment analysis are introduced by (Barnes et al., 2018). They performed tokenization, POS tagging and Lemmatization using Lxa-pipes. They trained a linear svc classifier for the classification of the polarity of opinion expressions. They trained a CRF on the standard features for the extraction of opinion holders, targets and expressions. For evaluation, 10 fold cross validation is used on 80% of data and F1 is used for extraction and classification.

(Apidianaki et al., 2016) this paper describes the data collection procedure and annotation guidelines. To increase the applicability and comparability of system, SemEval-2015 Task for English guidelines are used for annotation. They used different systems to identify three types of information (aspect category, opinion target expression and sentiment polarity). The categories returned by a framework are compared with gold annotations and Precision, Recall and F1 is measured.

(Nawaz et al., 2019) a segregational approach has been used for identifying the aspects. They introduced a technique which is consisted on two phases. 1st phase consist on extraction and grouping of target related words for a given objective by utilizing Normalized Google distance (NGD). Aspects are identified using POS tagger which is modified. In 2nd phase, they reduced the redundant and irrelevant aspects using Concept Net. They have applied this strategy on each word in an opinionated sentence to identify either it is a aspect word or non-aspect word.

(Brychcín et al., 2014) participated in the task of SemEval task 4 2014. They have used machine learning approach for constrained system and for unconstrained system they expand the constrained feature set by LDA, semantic spaces and semantic dictionaries. They compared their results with best and averages as well as with the baseline of SemEval and found that performance of their system is quite well.

(Zhao et al., 2014) worked on introducing a new annotation scheme and developed corpus for Chinese sentiment analysis. Elements of their annotation scheme are target entity, aspect, implicit aspect, polarity expression, modifier, Negation, polarity, transition word and compare. For sentiment analysis task, 3 main most relevant elements are ¡object, description, polarity¿. In their annotation scheme, 1st three elements are for object then 4-6 are for description and last three elements can be used to compute polarity. They performed an experiment on target-aspect pair extraction task. For cross validation, they used 10 fold on the new corpus. They applied Ml algorithms and reached the accuracy at 81.80%.

(Clematide et al., 2012) worked on the creation of Multilayered Reference corpus for German sentiment analysis. They used layered approach for annotation. At layer 1, polarity and subjectivity/objectivity is analyzed. At layer 2, they analyzed word and phrase-level annotation. At layer 3, they focused on the annotation of

expression-level. Each layer had been analyzed by multiple annotators. To ensure the quality of data, they also calculated the inter-annotator agreement.

(Kumar et al., 2018) worked on the development of Hindi-English corpus for aggression annotation. They defined an annotation scheme in which they divided aggression tag sets into three levels (top-level). Overtly aggressive, covertly aggressive and non aggressive. Each of the top level has further two attributes – discursive role and discursive effect. Discursive effects have 10 kind(s) and are based on the type of aggression. They used a hierarchical approach of 3 top-level tags and 10 level 2 tags for annotation.

(Kılınç et al., 2017) worked on the creation of dataset named as TTC 3600. Stemming is performed on the primary data. For feature selection, two methodologies are used. Correlation-based feature selection and Attribute ranking based feature selection. Thinking about the high dimensionality and over fitting characteristics five classifiers were selected for text categorization (NB, SVM, K-NN, j48 and RF). Three different versions of TTC 3600 (F5-DS, F7-DS and Zemb-DS) were created by removing stop words and using stemmer. RF has the highest accuracy on all the datasets as compared to other classifiers. RF gives the best accuracy 91.03% on Zemb-DS dataset after applying ARFS.

(Tocoglu and Alpkocak, 2018) a dataset for emotion analysis in Turkish language has been prepared by them named as TREMO. They performed a validation process to validate the raw dataset and got two datasets. In preprocessing, they performed Fixed prefix (F5) stemming and Zemberek stemming. Which resulted four versions of dataset, F5, F5-V, Z and Z-V. For feature selection, they used mutual information. Then they applied four different classifiers (CNB, J48, RF and SVM) on all types of datasets and evaluate the performance by Precision, Recall, accuracy and F-measure. Result of SVM was better than the others.

3 Statement of the problem

Rather than classifying the complete sentiment of a sentence into positive, negative or neutral, ABSA allows us to associate specific sentiment with different aspects of a product. Since Urdu language is a low-resourced language, there is very little work on ABSA, especially in Urdu language (ul Haq et al., 2020). Unavailability of Urdu dataset for aspect based sentiment analysis leads to constructing a benchmark dataset of a specific domain. We aim to create a benchmark corpus to facilitate the ABSA systems. Annotation will be performed according to the guidelines of SemEval and performance is measured by using different Machine Learning approaches.

4 Proposed Methodology

As our proposed methodology can be seen from figure 1.

We crawled Urdu tweets of Cricket and Football domain from Twitter using Twitter API's. Then we performed preprocessing on raw data. In preprocessing, special characters, hashtags, links and punctuation marks are removed using Sci Kit Learn and regular expressions. After preprocessing, we prepared annotation guidelines for annotating the data. These guidelines are according to the standards of SemEval. After preparing guidelines for annotation, three annotators will manually annotate the dataset. In the annotation process, four types of information have to identify, i.e., aspect, aspect category, aspect polarity and aspect category polarity. Annotators will identify aspects and assign polarity. To ensure the quality of the dataset, we will compute the inter-annotator agreement. For feature extraction, we will apply Tf-idf vectorizer and n-gram models. Next, we will apply Baseline models of Machine Learning like naive Bayes, Random Forest, KNN. Then we will evaluate the performance by Precision, Recall and F1 measure.

5 Data Collection and Annotation

This dataset is consists of two different domains "Cricket" and "football". We collected 7000 tweets of cricket domain and 3000 tweets of football domain from Twitter using Twitter API's. Three annotators will manually annotate the data. They have to identify four types of information, which is discussed in section 4.

5.1 Data description

This section describes the data statistics that have been annotated in the developed corpus. Table 1 shows the aspect category statistics.

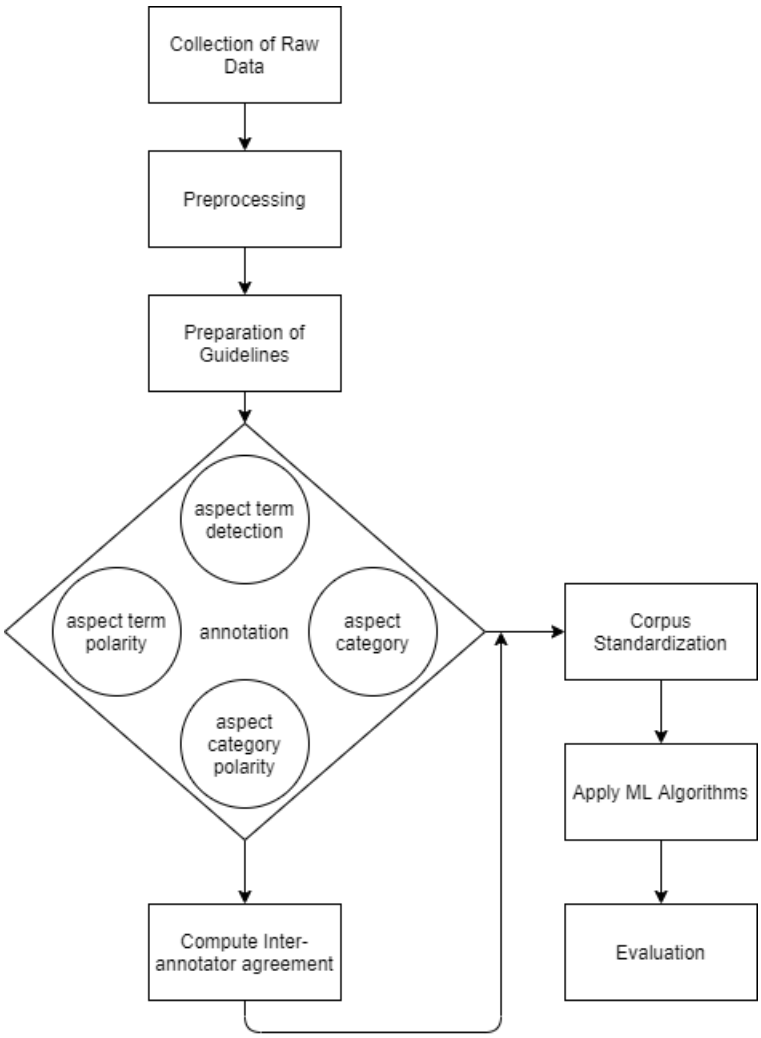


Figure 1: Research Methodology

6 Annotation guidelines

Initially, we have prepared some guidelines to annotate the data. The aim of these guidelines is to identify the aspects and sentiment polarity in sentences. We have to identify four types of information for annotation.

- Aspect / Entity
- Aspect Polarity
- Aspect Category
- Aspect Category Polarity

6.1 Guidelines for Aspect / Entity Extraction

Usually Noun / Nominal phrases indicate aspect term / aspect terms.

- Words / phrases that are expressing opinion (subjectivity indicator) are not considered aspect term.
- If an aspect term occurs more than once, then annotate all of them.
- If an identified aspect is misspelled, it should be annotated.
- Annotate aspect terms even if it is in quotation marks or brackets.
- Implicit aspects are not annotated.

6.2 Guidelines for Aspect polarity

Assign polarity to each aspect from these (positive, negative, neutral and conflict).

- When a sentence contains positive sentiment then assign positive polarity.
- When a sentence contains negative sentiment then assign negative polarity.
- Assign neutral polarity to aspect when an aspect is neither positive nor negative.

Category	Comments	Pos. Comments	Neg. Comments	Neutral Comments
performance	2117	1299	668	150
general	1678	653	496	529
inquiry Commission	89	14	58	17
management	1099	231	430	438
other	1689	313	1002	374
Total	6672	2510	2654	1508

Table 1: Statistics of aspect categories

- When a sentence has more than 1 aspect and each aspect have a contrast among polarities then assign conflict polarity to the aspects.

6.3 Guidelines for Aspect Category

Identify the aspect category from these predefined categories. Assign "other" category to those aspects which have implicit aspect.

- management
- performance
- inquiry commission
- general
- other

6.4 Guidelines for Aspect Category Polarity

It is same as Aspect term polarity. Assign the polarity from (positive, negative, neutral and conflict).

7 Conclusion

In this paper, we have presented a benchmark corpus for ABSA in the Urdu language. This dataset has been prepared to cover different areas of research. It will facilitate researchers because we are covering four tasks of sentiment analysis. Which can be further used in the future by adding more domains. We believe that it will be very beneficial for the researcher community because we work in a different language and a specific domain. In future, we aim to improve our work by applying different techniques for feature extraction. We also plan to explore more machine learning algorithms and neural networks for these tasks.

References

M. Al-Smadi, O. Qawasmeh, B. Talafha, and M. Quwaider. 2015. Human annotated arabic dataset of book reviews for aspect based sentiment analysis. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 726–730.

Marianna Apidianaki, Xavier Tannier, and Cécile Richart. 2016. Datasets for aspect-based sentiment analysis in french.

Huniya Arif, Kinza Munir, Abdul Subbooh Danyal, Ahmad Salman, and Muhammad Moazam Fraz. 2016. Sentiment analysis of roman urdu/hindi using supervised methods. *Proceedings of ICICC*, 8:48–53.

Jeremy Barnes, Patrik Lambert, and Toni Badia. 2018. Multibooked: A corpus of basque and catalan hotel reviews annotated for aspect-level sentiment classification. *arXiv preprint arXiv:1803.08614*.

Tomáš Brychcín, Michal Konkol, and Josef Steinberger. 2014. Uwb: Machine learning approach to aspect-based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 817–822.

Simon Clematide, Stefan Gindl, Manfred Klenner, Stefanos Petrakis, Robert Remus, Josef Ruppenhofer, Ulli Waltinger, and Michael Wiegand. 2012. Mlsa—a multi-layered reference corpus for german sentiment analysis.

Mazen El-Masri, Nabeela Altrabsheh, and Hanady Mansour. 2017. Successes and challenges of arabic sentiment analysis research: a literature review. *Social Network Analysis and Mining*, 7(1):54.

Darina Gold, Marie Bexte, and Torsten Zesch. 2018. Corpus of aspect-based sentiment in political debates.

Ehsan ul Haq, Sahar Rauf, Sarmad Hussain, and Kashif Javed. 2020. Corpus of aspect-based sentiment for urdu political data. *LANGUAGE & TECHNOLOGY*, page 37.

Khairullah Khan, W Khan, A Rehman, A Khan, and Asfandyar Khan. 2018a. Urdu sentiment analysis. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 9(9).

SS Khan, M Khan, Q Ran, and R Naseem. 2018b. Challenges in opinion mining, a comprehensive

review. *Sci. Technol. J.(Ciencia e Tecnica Vitivinicola)*, 33(11):123–135.

Deniz Kılınç, Akın Özçift, Fatma Bozyigit, Pelin Yıldırım, Fatih Yücalar, and Emin Borandag. 2017. Ttc-3600: A new benchmark dataset for turkish text categorization. *Journal of Information Science*, 43(2):174–185.

Ritesh Kumar, Aishwarya N Reganti, Akshit Bhatia, and Tushar Maheshwari. 2018. Aggression-annotated corpus of hindi-english code-mixed data. *arXiv preprint arXiv:1803.09402*.

Asif Nawaz, Sohail Asghar, and Syed Husnain Abbas Naqvi. 2019. A segregational approach for determining aspect sentiments in social media analysis. *The Journal of Supercomputing*, 75(5):2584–2602.

Priyanka Patil and Pratibha Yalagi. 2016. Sentiment analysis levels and techniques: A survey. *space*, 1:6.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Zia Ul Rehman and Imran Sarwar Bajwa. 2016. Lexicon-based sentiment analysis for urdu language. In *2016 sixth international conference on innovative computing technology (INTECH)*, pages 497–501. IEEE.

Afraz Z Syed, Muhammad Aslam, and Ana Maria Martinez-Enriquez. 2010. Lexicon based sentiment analysis of urdu text using sentiunits. In *Mexican International Conference on Artificial Intelligence*, pages 32–43. Springer.

Mansur Alp Tocoglu and Adil Alpkocak. 2018. Tremo: A dataset for emotion analysis in turkish. *Journal of Information Science*, 44(6):848–860.

Yanyan Zhao, Bing Qin, and Ting Liu. 2014. Creating a fine-grained corpus for chinese sentiment analysis. *IEEE Intelligent Systems*, 30(1):36–43.

Making a Point: Pointer-Generator Transformers for *Disjoint* Vocabularies

Nikhil Prabhu and **Katharina Kann**
University of Colorado Boulder
{nikhil.prabhu,katharina.kann}@colorado.edu

Abstract

Explicit mechanisms for copying have improved the performance of neural models for sequence-to-sequence tasks in the low-resource setting. However, they rely on an overlap between source and target vocabularies. Here, we propose a model that does not: a pointer-generator transformer for *disjoint* vocabularies. We apply our model to a low-resource version of the grapheme-to-phoneme conversion (G2P) task, and show that it outperforms a standard transformer by an average of 5.1 WER over 15 languages. While our model does not beat the the best performing baseline, we demonstrate that it provides complementary information to it: an oracle that combines the best outputs of the two models improves over the strongest baseline by 7.7 WER on average in the low-resource setting. In the high-resource setting, our model performs comparably to a standard transformer.

1 Introduction

Deep learning models define the state of the art on the majority of sequence-to-sequence tasks in natural language processing (NLP). Even when training data is limited, neural networks outperform many alternative approaches, e.g., on machine translation (Sennrich and Zhang, 2019) or on morphological generation tasks (Cotterell et al., 2018). For the second group, the use of mechanisms for copying has drastically improved performance when training sets are small (Cotterell et al., 2018).

Our work builds on the insight that the ability to copy elements from the input over to the output – as done by a pointer network (Vinyals et al., 2015) or a pointer-generator network (See et al., 2017) – can increase model performance on sequence-to-sequence tasks in the low-resource setting, as it simplifies the learning problem. However, existing neural models require that inputs and outputs consist of elements from overlapping sets. Here, we propose a **pointer-generator transformer model for *disjoint* source and target vocabularies**. Our model, shown in Figure 1, is a hybrid of an LSTM pointer-generator model (See et al., 2017) and a transformer model (Vaswani et al., 2017). Additionally, we integrate a mapping function, which defines a correspondence between elements in the source and target vocabularies.

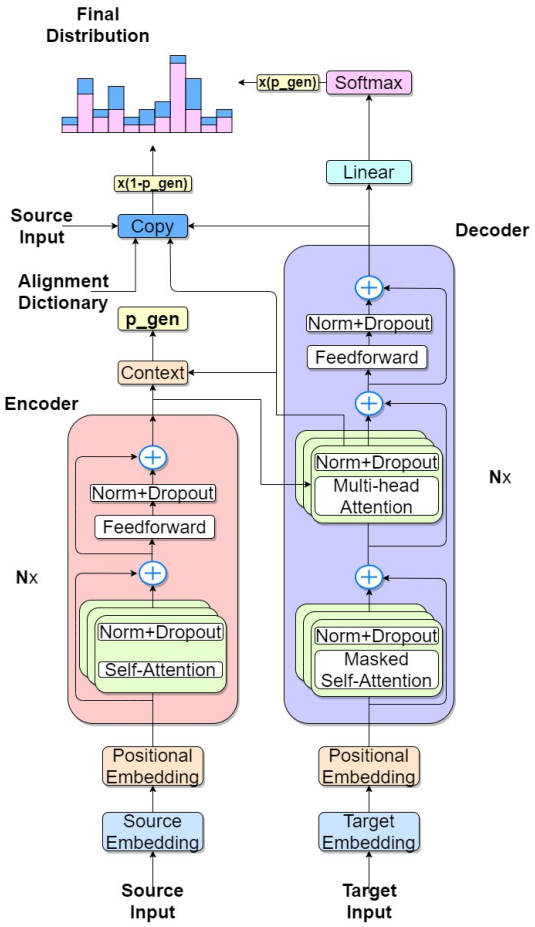


Figure 1: Architecture of our pointer-generator transformer for disjoint vocabularies.

We apply our model to the task of grapheme-to-phoneme conversion: mapping the spelling of

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 78–85
December 4 - 7, 2020.

a word to a representation of its pronunciation (Bisani and Ney, 2008). This task is a great testing ground for our approach: input and output vocabularies are disjoint for many languages (cf. Table 1), and, as a character-level task with short sequences, it enables us to use small models, which allows for quick experimentation. G2P is also a task of high practical relevance: it is required for text-to-speech synthesis. Models that perform well in the low-resource setting will enable us to develop such language technologies for a wider set of languages.

We experiment with our model on varying training set sizes. In the low-resource setting, averaged over 15 languages, our architecture improves performance by 5.1 WER over a standard transformer. Further, it outperforms both the transformer and a copy baseline for up to 1000 training examples. While it underperforms our best performing baseline, we show that it provides valuable complementary information to it. In the high-resource setting, it performs comparably to a vanilla transformer.

2 Related Work

Sequence-to-sequence models. Popular neural architectures for sequence-to-sequence tasks include those based on LSTMs or GRUs in combination with attention (Bahdanau et al., 2015), transformer models, which use attention instead of recurrence (Vaswani et al., 2017), or pointer-generator models based on LSTMs (See et al., 2017). Sequence-to-sequence models have been applied to a large set of NLP tasks, including translation (Bahdanau et al., 2015; Vaswani et al., 2017), summarization (Raffel et al., 2019), morphological generation (Kann and Schütze, 2016), or historical text normalization (Flachs et al., 2019). To the best of our knowledge, pointer-generators have so far only been applied to tasks with overlappings source and target vocabularies (See et al., 2017; Sharma et al., 2018; Deaton et al., 2019). Here, we propose a pointer-generator transformer for tasks with *disjoint* vocabularies.

G2P. Early algorithms for G2P relied on handwritten parser-based rules in the format of Chomsky-Halle rewrite – or LTS – rules (Chomsky and Halle, 1968). Subsequently, other techniques have been developed, including rule-based systems (Black et al., 1998), maximum entropy models (Chen, 2003), LSTMs (Rao et al., 2015), or approaches based on semi-automatic alignment tables (Pagel et al., 1998). Our approach is similar to the idea of alignment tables, since we integrate a mapping function between vocabularies into a pointer-generator transformer. Today, neural sequence-to-sequence models are the standard approaches for G2P (Yao and Zweig, 2015; Sun et al., 2019; Gorman et al., 2020).

Makarov and Clematide (2020) proposed a model for G2P that, similar to our approach, makes use of explicit substitutions. Their BiLSTM-based neural transducer learns edit actions to transform an input sequence into a target sequence and is trained with imitation learning. Since this model is highly suitable for the low-resource setting, we compare our approach to it in our experiments.

	Low-Resource			High-Resource		
	Σ_{src}	Σ_{trg}	$\cap$	Σ_{src}	Σ_{trg}	$\cap$
arm	38	38	0	38	58	0
bul	29	40	0	30	67	0
fre	28	33	18	37	40	21
geo	31	32	0	33	35	0
gre	32	31	1	38	44	1
hin	49	48	0	60	88	0
hun	32	53	20	34	70	21
ice	31	53	19	36	77	23
kor	165	51	0	834	61	0
lit	31	64	18	32	110	20
ady	30	62	0	32	105	0
dut	27	40	18	35	50	21
jpn	65	47	0	78	79	0
rum	26	39	18	52	71	22
vie	75	47	15	93	49	18
avg.	**45.93**	**45.2**	**8.47**	**97.47**	**66.93**	**9.8**

Table 1: Number of tokens in the source and target vocabularies, and the number of shared tokens.

3 Pointer-Generator Transformers for Disjoint Vocabularies

Hyperparameter	Value
Batch Size	128
Embedding Dimension	256
Hidden Dimension	1024
Dropout	0.3
Number of Encoder Layers	4
Number of Decoder Layers	4
Number of Attention Heads	4
Learning Rate	1e-3
β_1	0.9
β_2	0.998
Label Smoothing Coefficient	0.1
Max Norm (Gradient clipping)	1

Table 2: The hyperparameters used in our experiments.

	Low-resource						High-resource					
	PG-T	T	LSTM	IM	Sub	O*	PG-T	T	LSTM	IM	Sub	O*
arm	62.2	70.5	72.2	**51.6**	57.3	38.2	15.6	**14.5**	14.7	14.9	57.3	10.7
bul	84.7	87.1	78.0	**73.6**	90.0	63.6	33.1	31.9	31.1	**29.8**	86.9	18.7
fre	87.1	88.7	93.1	**68.9**	95.3	61.3	7.3	8.0	**6.2**	7.6	95.3	4.7
geo	65.6	77.8	60.2	62.0	**50.7**	45.1	**25.6**	27.8	26.4	26.9	44.7	19.3
gre	77.6	81.1	84.2	**44.0**	74.2	39.6	**17.1**	18.0	18.9	18.2	71.1	11.1
hin	**70.7**	82.6	80.4	78.0	72.7	60.2	8.0	8.2	**6.7**	6.9	69.8	3.8
hun	74.0	83.2	83.1	**41.1**	56.2	36.9	6.7	5.7	5.3	**4.4**	54.9	3.1
ice	87.1	90.9	94.2	**60.0**	85.3	56.0	11.1	10.6	**10.0**	11.6	85.3	8.2
kor	99.6	98.9	100.0	**75.8**	100.0	75.6	35.1	33.5	46.9	**28.7**	100.0	20.2
lit	87.1	89.6	97.1	**58.7**	97.6	52.9	22.2	21.0	19.1	**18.2**	96.9	16.2
ady	87.1	90.8	89.3	**64.0**	90.9	58.9	**27.1**	27.6	28.0	30.4	91.3	22.5
dut	90.9	93.0	93.3	**61.1**	96.2	58.5	18.7	18.2	**16.4**	19.8	97.1	12.5
jap	77.8	85.7	97.3	**76.2**	99.8	64.0	7.3	7.4	7.6	**7.1**	99.6	5.3
rum	64.0	73.5	59.3	**33.1**	48.9	29.1	13.6	12.1	**10.7**	13.8	51.3	10.9
vie	90.0	88.3	99.6	**82.7**	100.0	75.8	4.4	3.6	4.7	**1.1**	100.0	1.1
Avg.	80.4	85.5	85.4	**62.1**	81.0	54.4	16.9	16.5	16.9	**16.0**	80.1	11.2

Table 3: Test set results for WER; all models are described in the text. The best performance (excluding the oracle) is shown in bold.

3.1 Model Architecture

Our model, cf. Figure 1, is a hybrid of a transformer (Vaswani et al., 2017) and a pointer-generator network (See et al., 2017) with a separate mapping function between vocabularies. Like the transformer, it is an auto-regressive encoder-decoder architecture with stacked self-attention and fully connected encoder and decoder layers. The decoder, in addition, employs multi-head attention over the encoder outputs. We further add a component which outputs the probability of generating a token, as opposed to copying an element from the source. Following See et al. (2017), the generation probability $p_{gen} \in [0, 1]$ at time step t is computed as:

$$p_{gen} = \sigma(w_c^T c_t + w_s^T s_t + w_x^T x_t + b_{ptr}) \quad (1)$$

c_t represents the context vector at step t, which is the sum of the embedded encoder hidden states h, weighted by the multi-head attention weights a: $c_t = \sum_i a_i^t h_i$. s_t and x_t are the hidden respresentation in the last decoder layer and, respectively, the decoder input. $w_{h^*}^T$, w_s^T, w_x^T are learned weights and b_{ptr} is a learned bias vector.

We also account for the fact that our architecture should handle *disjoint* source and target vocabularies, denoted as as Σ_{src} and Σ_{trg}. To enable the use of a pointer mechanism across vocabularies, we define a mapping function m from the source vocabulary to the target vocabulary:

$$m : \Sigma_{src} \rightarrow \Sigma_{trg} \quad (2)$$

Our model then computes the probability of an output character $k \in \Sigma_{trg}$ for an input sequence $g \in \Sigma_{src}^*$ at each time step t as:

$$P(k) = p_{gen} P_{\Sigma_{trg}}(k) + (1 - p_{gen}) \sum_{i:k=m(g_i)} a_i^t \quad (3)$$

$P_{\Sigma_{trg}}(k)$ represents the probability of k to be generated by the decoder in the standard way, given the input sequence and previously generated tokens. It is weighted by the generation probability p_{gen}. The second term represents the attention over the encoder outputs, which is multiplied by $1 - p_{gen}$, the copy probability. The target indices receiving probability are found with the mapping m.

3.2 Source-to-Target Vocabulary Mapping

We obtain the mapping function m as follows.

Alignment. We first compute an alignment between source and target tokens in the training files for all languages and settings with the GIZA++ aligner (Och and Ney, 2003), employing the default parameters. The alignment is computed by a hidden Markov model.

Mapping. We then construct a mapping between characters in the source and target vocabularies by assigning the target token with the highest type-level alignment probability to each source token. The mapping function m can map multiple source tokens to the same target token. We create separate mapping functions for all languages and settings in our experiments as they change for different training sets.

	Low-resource					High-resource				
	PG-T	T	LSTM	IM	Sub	PG-T	T	LSTM	IM	Sub
arm	19.0	23.6	21.3	14.4	**12.3**	3.9	3.5	3.5	**3.3**	12.1
bul	30.9	33.6	23.1	**19.2**	25.6	7.1	6.5	5.9	**5.7**	25.7
fre	40.2	44.1	41.9	**22.0**	53.7	1.9	2.0	**1.3**	1.8	52.8
geo	18.9	24.0	13.3	11.8	**8.7**	4.8	5.2	5.1	**4.6**	8.2
gre	26.7	29.2	24.0	**10.3**	18.5	**2.8**	2.9	3.3	3.2	18.0
hin	26.9	35.7	31.6	29.2	**23.4**	2.0	2.0	**1.5**	1.4	22.1
hun	24.6	33.1	27.4	**10.0**	17.8	1.4	1.3	**1.2**	**1.2**	17.5
ice	43.0	44.7	42.7	**17.9**	30.4	2.8	**2.4**	**2.4**	2.5	29.6
kor	65.1	69.0	78.3	**23.8**	52.7	10.4	9.9	16.8	**4.7**	50.4
lit	39.5	43.2	53.1	**8.8**	37.4	4.2	3.9	3.6	**2.9**	36.9
ady	37.3	42.1	33.1	**18.2**	46.9	6.8	6.6	**6.5**	7.2	44.8
dut	42.6	44.5	40.8	**15.1**	37.0	3.6	3.6	**2.9**	3.7	37.0
jap	34.1	41.4	54.6	**26.1**	48.4	1.9	2.2	1.8	**1.7**	50.6
rum	21.4	26.0	14.1	**8.5**	11.4	3.1	2.7	**2.5**	3.7	10.7
vie	44.5	45.1	65.1	**27.2**	65.4	1.5	1.6	1.5	**0.3**	58.8
Avg.	34.3	38.6	37.6	**17.5**	32.6	3.9	3.8	4.0	**3.2**	31.7

Table 4: Test set results for PER; all models are described in the text. The best performance is shown in bold.

Our system is built on the transformer implementation by Wu et al. (2020), and our final code is available on github.[1]

4 Experiments

4.1 Data, Metrics, and Baselines

Data. We use the G2P data from Gorman et al. (2020), which covers a set of 15 typologically diverse languages. We construct our low-resource experiments by taking the first 100 instances from each training set. For the high-resource experiments, we leverage all available data. Development and test sets are the same in both settings. Table 1 shows the number of characters in Σ_{src} and Σ_{trg} according to the training set, as well as the number of characters shared between both. We can see that, for many languages, the overlap is 0, i.e., the vocabularies are disjoint sets.

Metrics. We use word error rate (WER), i.e., the percentage of words that are correct, as our main metric. We also measure the phoneme error rate (PER), i.e., the percentage of correctly generated phonemes. We use the SIGMORPHON 2020 Task 1 (Gorman et al., 2020) evaluation script[2] to calculate these.

Baselines. We compare our model (**PG-T**) to four baselines: **T** is a vanilla transformer with hyperparameters identical to those in PG-T for the high-resource setting (following Gorman et al. (2020); listed in Table 2). For the low-resource setting, all hyperparameters remain the same, except the batch size which is set to 32. Thus, T corresponds to our model, but without the pointing mechanism. **IM** is the neural transducer trained with imitation learning detailed in Section 2 (Makarov and Clematide, 2020). **LSTM** is the LSTM encoder-decoder baseline of SIGMORPHON 2020 Task 1 (Gorman et al., 2020). The output of **Sub** is the sequence of target tokens corresponding to each source token, according to m. Finally, **O*** is not a baseline but an oracle model that combines the best outputs of **PG-T** and **IM**.

4.2 Results

Low-resource G2P. Table 3 shows the WER of our model and all baselines on the test set. In the low-resource setting, PG-T outperforms all baselines except for IM, with an average increase of 0.6 over Sub, the second best baseline after IM. It further improves over T by 5.1 WER, which shows the effectiveness of our pointer extension. While the average improvement over Sub is modest, PG-T shows large performance gains over Sub for individual languages such as Japanese. Similarly, for Hindi, our model outperforms IM by 7.3 WER.

High-resource G2P. In the high-resource setting, PG-T obtains an average performance of 16.9 WER. T and IM slightly outperform PG-T: T by 0.4 WER and IM by 0.9 WER. PG-T and LSTM reach the same average WER. Sub improves minimally as compared to the low-resource setting and is vastly outperformed by all other approaches. We show PER results for the low-resource and the high-

[1] https://github.com/nala-cub/g2p-PG-T

[2] https://github.com/sigmorphon/2020/blob/master/task1/evaluation/evaluate.py

resource setting in Table 4. All development set results for both WER and PER can be found in the appendix.

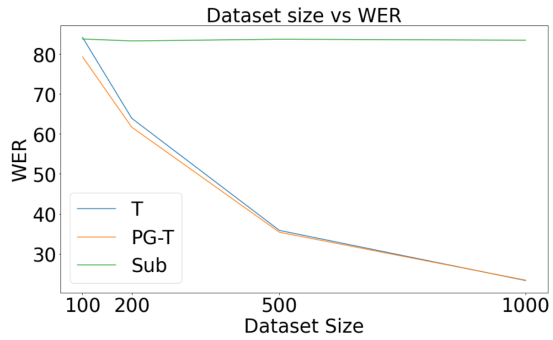


Figure 2: Performance for increasing amounts of training data on the development set.

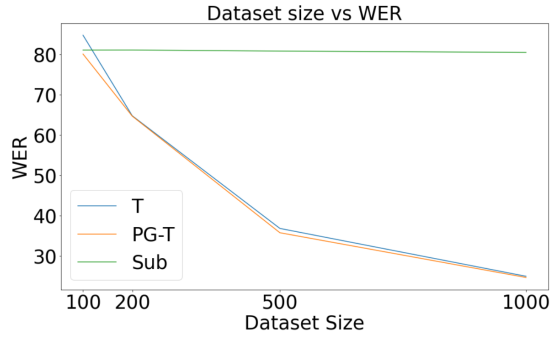


Figure 3: Performance for increasing amounts of training data on the test set.

Learning curve. We also look at the learning curves for increasing dataset sizes (Figures 2, 3) and compare our model to the two baselines T and Sub, whose main ideas PG-T combines. Compared to T, PG-T shows the biggest improvements for 100 training examples. The performance difference between the two gets smaller as the size of the dataset increases. However, for 1000 training examples, PG-T still performs slightly better than T. With regards to Sub, our model outperforms it slightly for 100 examples, but this gap widens very quickly, due to Sub's inability to learn much from the training data. Overall, we show that, over a varied amount of data in the low-resource setting, our model outperforms both baselines and, thus, that our combination of the two is effective.

While our model does not perform as well as IM, our strongest baseline, we demonstrate that it provides valuable complementary information. In both the low-resource and the high-resource setting and across all languages, an oracle combination of PG-T with IM yields better results than either model on its own. We will discuss this in the next section.

	CCO	UCO-PG-T	UCO-IM	CMO
arm	115	69	78	127
bul	40	23	146	52
fre	36	42	122	43
geo	91	59	124	118
gre	64	28	158	77
hin	48	79	54	53
hun	122	22	175	131
ice	47	17	132	56
kor	2	2	104	3
lit	33	15	168	39
ady	37	18	128	45
dut	26	10	158	30
jpn	40	51	61	43
rum	150	16	144	165
vie	21	34	77	22
Avg.	**58.13**	**32.33**	**121.93**	**66.9**

Table 5: Output comparison of PG-T and IM for the development set in the low-resource setting.

	CCO	UCO-PG-T	UCO-IM	CMO
arm	354	21	19	397
bul	257	43	34	294
fre	385	16	19	401
geo	299	38	35	365
gre	349	19	34	385
hin	403	14	15	418
hun	425	5	11	431
ice	384	13	20	403
kor	288	28	69	339
lit	326	14	38	364
ady	313	28	26	376
dut	346	28	27	378
jpn	410	7	8	430
rum	380	11	14	419
vie	422	5	19	425
Avg.	**356.1**	**19.3**	**25.87**	**388.33**

Table 6: Output comparison of PG-T and IM for the development set in the high-resource setting.

5 Analysis

5.1 Oracle

O* in Table 3 is a hypothetical oracle that would be the result of combining the best outputs of PG-T and IM. The results in Table 3 show that the oracle performs far better than IM, by 7.7 WER and 4.8 WER in the low-resource and the high-resource setting, respectively. This indicates that there are differences in the behaviour of the two models, and that they provide complementary information. We will discuss the differences in the models' predictions next.

5.2 Output Comparisons

In this subsection, we go over the output comparisons in Tables 5 and 6. In those tables, we compare the phoneme outputs generated by PG-T and IM. In particular, we look at the number of (1) correct outputs both models have in common (CCO), (2) correct outputs that are unique to each model (UCO-PG-T and UCO-IM), and (3) outputs both models have in common (CMO), independent of if they are correct or incorrect.

Tables 5 and 6 compare the outputs of PG-T and IM in the low-resource and the high-resource setting, respectively. The average CCO count is 58.13 in the low-resource setting and 356.1 in the high-resource setting (out of a total of 450 data points), showing that part of what is learned is shared between both models. However, in the high-resource setting, the average UCO counts are 19.3 for UCO-PG-T and 25.87 for UCO-IM, respectively. This indicates that, while their performances are similar, the models are learning some complementary information. This is reflected in the low-resource setting as well: even though IM performs better on average, PG-T still learns relevant complementary information, with an average of 32.33 UCO-PG-T. We find the same for the individual languages in our experiments.

5.3 Error Analysis

As a case study, we further perform an analysis of the errors on the Hindi development set in the high-resource setting.[3] For this particular combination of language and dataset size, LSTM performs best with 4.7 WER, followed by IM with 7.1 WER, PG-T with 7.3 WER, and T with 8.6 WER. Sub performs badly with 68.2 WER. Almost all errors generated by the high-performing models (IM, LSTM, PG-T and T) center around the halant[4] morpheme in the Hindi script, which indicates the lack of an inherent vowel following a consonant. Its phoneme complement is the schwa[5], which indicates an unstressed vowel. LSTM and T behave identically on most examples, while PG-T often behaves in the opposite way, i.e., in cases where LSTM generates a schwa, PG-T does not, and vice versa. PG-T and IM, however, have an even split among examples where either model makes a schwa-based error.

[3] WER and PER of all models on the development sets for all languages are shown in Table 7 in the appendix.

[4] https://en.wiktionary.org/wiki/halant

[5] https://en.wiktionary.org/wiki/schwa

6 Conclusion

We introduced a pointer-generator transformer for sequence-to-sequence tasks with disjoint vocabularies, and evaluated it on G2P in 15 different languages. While our model did not perform as well as our strongest baseline, we showed that our model learns complementary information to the latter, thus succeeding on examples the baseline fails on. We further demonstrated that combining the best outputs of our pointer-generator transformer and the baseline results in a lower WER on low-resource G2P than each individual model. We leave the development of a system that can combine the two G2P systems for future work.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*. International Speech Communication Association.

Stanley F Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eighth European Conference on Speech Communication and Technology*.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English*. Harper & Row New York.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Jon Deaton, Austin Jacobs, Kathleen Kenealy, and Abigail See. 2019. Transformers and pointer-generator networks for abstractive summarization.

Simon Flachs, Marcel Bollmann, and Anders Søgaard. 2019. Historical text normalization with delayed rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1614–1619, Florence, Italy. Association for Computational Linguistics.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya D. McCarthy, Shijie Wu, and Daniel You. 2020. The sigmorphon 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2020. Cluzh at sigmorphon 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 171–176.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Vincent Pagel, Kevin Lenzo, and Alan Black. 1998. Letter to sound rules for accented lexicon compression. *arXiv preprint cmp-lg/9808010*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229. IEEE.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. *arXiv preprint arXiv:1905.11901*.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 105–111, Brussels. Association for Computational Linguistics.

Hao Sun, Xu Tan, Jun-Wei Gan, Hongzhi Liu, Sheng Zhao, Tao Qin, and Tie-Yan Liu. 2019. Token-level ensemble distillation for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1904.03446*.

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.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the transformer to character-level transduction. *arXiv preprint arXiv:2005.10213*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Appendix B: Development Set Results

	Low-resource						High-resource					
	PG-T	T	LSTM	IM	Sub	O*	PG-T	T	LSTM	IM	Sub	O*
arm	59.1	70.7	74.0	57.1	**56.2**	41.8	16.5	16.0	14.9	17.1	**56.7**	3.8
bul	86.0	88.4	84.7	**58.7**	92.7	53.6	33.3	32.3	**28.4**	35.3	90.4	25.8
fre	82.7	86.6	92.7	**64.9**	97.6	55.6	10.9	10.2	**7.1**	10.2	97.3	6.7
geo	66.7	76.4	63.6	**52.2**	56.4	39.1	25.1	26.1	**21.1**	25.8	51.1	17.3
gre	79.6	82.4	82.7	**50.7**	75.3	44.5	18.2	18.1	**13.6**	14.9	74.7	10.7
hin	**71.8**	78.7	79.8	77.3	72.2	59.8	7.3	8.6	**4.7**	7.1	68.2	4.0
hun	68.0	81.2	80.9	**34.0**	75.1	29.1	4.4	4.4	3.3	**3.1**	74.7	2.0
ice	85.8	89.1	93.3	**60.2**	93.1	56.5	11.8	11.8	**9.3**	10.2	93.1	7.3
kor	99.1	99.4	100.0	**76.4**	100.0	76.0	29.8	26.5	41.1	**20.7**	100.0	14.5
lit	89.3	90.2	97.8	**55.3**	98.0	52.0	24.4	22.7	**16.9**	19.1	98.0	16.0
ady	87.8	88.0	88.7	**63.3**	91.3	59.3	24.2	24.6	**22.7**	24.7	90.4	18.5
dut	92.0	92.4	94.4	**59.1**	96.4	56.9	16.9	16.6	**12.2**	17.1	99.1	10.9
jap	79.8	85.0	94.7	**77.6**	99.3	66.2	7.3	7.0	**6.7**	7.1	100.0	5.6
rum	63.1	70.1	61.3	**34.7**	52.9	31.1	13.1	12.6	**9.3**	12.4	54.9	10.0
vie	87.8	88.1	98.4	**78.2**	100.0	70.7	5.1	4.5	4.2	**2.0**	100.0	0.9
Avg.	79.9	84.5	85.8	**60.0**	83.8	52.8	16.6	16.1	**14.4**	15.1	83.2	10.3

Table 7: Development set results for WER; all models are described in the text. The best performance (excluding the oracle) is shown in bold.

	Low-resource					High-resource				
	PG-T	T	LSTM	IM	Sub	PG-T	T	LSTM	IM	Sub
arm	18.4	23.5	22.5	14.2	**12.6**	3.4	3.4	**2.9**	3.4	12.6
bul	31.4	35.4	26.4	**15.9**	25.4	7.6	7.2	**6.2**	7.3	25.1
fre	40.9	44.8	44.6	**21.6**	58.2	2.8	2.6	**1.8**	2.9	57.5
geo	19.8	23.8	14.7	11.6	**10.1**	4.9	5.2	**4.5**	4.6	9.8
gre	27.5	29.3	24.6	**11.1**	20.4	3.6	3.4	**2.7**	**2.7**	20.6
hin	28.6	35.6	32.8	29.0	**23.3**	2.1	2.3	**1.4**	1.6	22.0
hun	23.3	32.5	26.8	**8.4**	22.7	0.9	1.0	**0.6**	0.7	22.4
ice	44.1	44.8	42.6	**17.6**	33.5	3.1	2.8	**2.0**	2.4	31.1
kor	68.0	72.2	78.5	**24.3**	53.1	7.6	7.6	16.6	**3.7**	51.2
lit	40.7	42.4	56.0	**8.7**	46.6	4.6	4.4	3.4	**3.1**	45.7
ady	37.9	41.9	32.6	**19.9**	48.0	6.0	6.4	**5.7**	5.9	45.0
dut	42.3	45.4	41.7	**14.7**	39.7	3.4	3.5	**2.1**	3.4	43.0
jap	34.0	40.0	55.0	**28.8**	48.4	2.5	2.4	2.1	**1.9**	49.8
rum	21.1	25.4	15.0	**9.2**	12.4	3.3	3.3	**2.3**	3.1	11.9
vie	46.5	44.4	63.8	**26.7**	66.9	1.5	1.6	1.3	**0.4**	63.7
Avg.	35.0	38.8	38.5	**17.5**	34.8	3.8	3.8	3.7	**3.1**	34.1

Table 8: Development set results for PER; all models are described in the text. The best performance is shown in bold.

Training with Adversaries to Improve Faithfulness of Attention in Neural Machine Translation

Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar
Simon Fraser University
8888 University Drive
Burnaby, BC, Canada
{pooya_moradi, nkambhat, anoop}@sfu.ca

Abstract

Can we trust that the attention heatmaps produced by a neural machine translation (NMT) model reflect its true internal reasoning? We isolate and examine in detail the notion of faithfulness in NMT models. We provide a measure of faithfulness for NMT based on a variety of stress tests where model parameters are perturbed and measuring faithfulness based on how often the model output changes. We show that our proposed faithfulness measure for NMT models can be improved using a novel differentiable objective that rewards faithful behaviour by the model through probability divergence. Our experimental results on multiple language pairs show that our objective function is effective in increasing faithfulness and can lead to a useful analysis of NMT model behaviour and more trustworthy attention heatmaps. Our proposed objective improves faithfulness without reducing the translation quality and it also seems to have a useful regularization effect on the NMT model and can even improve translation quality in some cases.

1 Introduction

Can we trust our neural models? This question has led to a wide variety of contemporary NLP research focusing on (a) different axes of interpretability including *plausibility* (or interchangeably *human-interpretability*) (Herman, 2017; Lage et al., 2019) and *faithfulness* (Lipton, 2018; Jacovi and Goldberg, 2020b), (b) interpretation of the neural model components (Belinkov et al., 2017; Dalvi et al., 2017; Vig and Belinkov, 2019), (c) explaining the decisions made by neural models to humans (using explanations, highlights, rationales, etc.) (Ribeiro et al., 2016; Li et al., 2016; Ding et al., 2017; Ghaeini et al., 2018; Bastings et al., 2019; Jain et al., 2020), and (d) evaluating different explanation methods from different perspectives

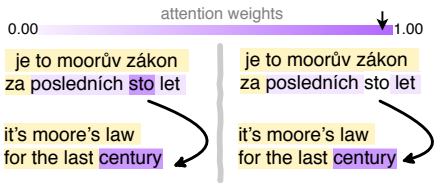


Figure 1: An example translation from Cs-En producing *unfaithful* attention weights. The model is generating the token century. In the left attention heatmap, the attention is on the word sto while the decoder generates century. However, in the right heatmap, sto is not attended to at all but century is still produced as the output. This is an example of unfaithful behavior. Yellow words are not attended.

(Samek et al., 2016; Mohseni and Ragan, 2018; Poerner et al., 2018; Jain and Wallace, 2019; Serrano and Smith, 2019; Wiegreffe and Pinter, 2019; Li et al., 2020).

Jacovi and Goldberg (2020b) emphasize distinguishing faithfulness from human-interpretability in interpretability research by providing several clarifications about the terminology used by researchers. They describe the following conditions on the evaluation of how well a research project tackles the notion of faithfulness: (1) Be explicit: provide a measurable evaluation of faithfulness, (2) Human judgements are not relevant because we are interested in model internals, (3) Do not match against gold labels (e.g. AER) because faithfulness of both correct and incorrect decisions made by the model are equally important, (4) No model is "inherently" faithful. We need to measure faithfulness not as a binary aspect of a model but rather as a gray-scale measure.

Aligned with these criteria, we study faithfulness of attention in NMT, the extent to which it can reflect the true internal reasoning behind a prediction (Figure 1). We make the following contributions:

- We propose a measure for quantifying faithfulness in NMT.
- We introduce a novel learning objective based

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 86–93
December 4 - 7, 2020.

on probability divergence that rewards faithful behavior and which can be included in the training objective for NMT.

- We provide empirical evidence that we can improve faithfulness in an NMT model. Our approach results in more a more faithful NMT model while producing better BLEU scores.

We chose to study the impact of faithfulness in NMT because it is under-studied in terms of interpretability. Most previous work has focused on document or sentence-based classification tasks where attention models are not as directly useful as in NMT models. Attention is also more challenging in terms of faithfulness in the context of NMT models due to the substantial impact of the decoder component.[1]

2 Faithfulness in NMT Models

Intuitively, a faithful explanation should reflect the true internal reasoning of the model. Although there is no formal definition for faithfulness, a common approach in the community is to design stress tests to perturb the model parameters chosen in such a way that the model's decision should change if the model is faithful (Jacovi and Goldberg, 2020b). A common stress test is the *erasure* test in which the most-relevant part of the input is removed (Arras et al., 2017). In the context of NMT, at decoding time step t the attention component assigns attention weights α_t, attending to the source word at position $m_t = \text{argmax}_i\, \alpha_t[i]$ (or the k-best attended-to words in the source). These weights are often implicitly or explicitly regarded as an interpretation for the model's prediction at the time step t (Tu et al., 2016; Mi et al., 2016; Liu et al., 2016; Wang et al., 2016; Lee et al., 2017; Ding et al., 2017; Ghaeini et al., 2018). The erasure stress test for evaluating faithfulness offered by α_t is done by setting $\alpha_t[m_t]$ to zero and observing whether or not the output changes. It is worth noting that erasure is only one of the possible stress tests for evaluating faithfulness. Passing more stress tests implies a more faithful model as it is resilient to more attacks or stress tests of its faithfulness. In this paper we consider three intuitive stress test cases:

- **ZeroOutMax** (Arras et al., 2017): Here we remove attention from the most important token according to the attention weights by setting $\alpha_t[m_t] = 0$.
- **Uniform** (Moradi et al., 2019): In this stress test all attention weights are set to be equal, $\alpha_t = \frac{1}{m}\vec{1}$, where m is the length of the source sentence. This is to confuse the model about which part of the input is the most important one.
- **RandomPermute** (Jain and Wallace, 2019): The attention weights are randomly permuted until a change in the model output is observed. We ensure that m_t, the most important token according to attention, is always changed. We set $\alpha'_t = \texttt{random\_permute}(\alpha_t)$ such that $\text{argmax}_i\, \alpha'_t[i] \neq m_t$

Many prior studies of attention (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019) have used a binary measure: either attention is faithful or it is not. These studies typically are about whether attention has the potential to be useful in terms of accuracy and faithful in terms of model behaviour. In many cases, especially in the case of NMT models, attention is clearly useful and by and large it must be faithful. The question is can we measure the faithfulness and improve faithfulness. It is more natural to have a gray-scale notion of faithfulness for evaluation (Jacovi and Goldberg, 2020b). Following this reasoning, we define $F(M)$ as faithfulness of attention heatmaps in model M as the following equation:

$$F(M) = \frac{\text{\# of tokens passing stress tests}}{\text{\# of tokens}} \quad (1)$$

$F(M)$ is a number between 0 to 1 measuring the percentage of output tokens during inference which passed the stress tests. This metric can also be regarded as a measure of trust we can assign to the attention heatmap to fully reflect the internal reasoning of the NMT model.

3 Approach

The conventional objective function in a sequence-to-sequence task is a cross-entropy loss $\mathcal{F}_{acc}$:

$$\mathcal{F}_{acc}(\theta) = \frac{1}{|S|} \sum_{(X,Y)\in S} \log p(Y|X;\theta) \quad (2)$$

[1] We focus on RNN based encoder-decoder models. While Transformers (Vaswani et al., 2017) generally produce better NMT models, in order to replace the long distance dependencies in a gated RNN, a Transformer model relies on multiple heads of attention and self-attention. Before we can tackle multi-head attention, we focus on the simpler single-head attention models and try to understand them in terms of faithfulness.

where S is the training data and X and Y are source sentence and the correct translation respectively. This training objective does not explicitly model the interpretability aspects (e.g. faithfulness) of the network and it remains unoptimized during training.

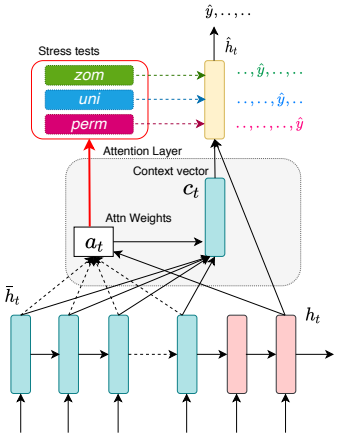


Figure 2: We generate adversaries to the attention weights using various stress tests *Uniform, ZeroOutMax,* and *RandomPermute*. When adversarial attention weights are used, in a faithful model we expect the probability of the original output ($\hat{y}$) to drop significantly. We use this criteria to define a faithfulness objective function.

Faithfulness Objective In an effort to develop a model that is *right for right reason*, Ross et al. (2017) change the loss function of their classifier to model both *right answers* and *right reasons* instead of only the former. They achieve this by introducing a regularizing term that tends to shrink irrelevant gradients. In a similar spirit, we change our objective to account for the NMT model's faithfulness as well as the cross-entropy score against the reference translations:

$$\mathcal{F} = \mathcal{F}_{acc} + \lambda_{faith}\mathcal{F}_{faith} \tag{3}$$

$\mathcal{F}_{faith}$ is an additional component that rewards the model for having more faithful attention. The parameter λ_{faith} regulates the trade-off between between faithfulness and accuracy objectives.

3.1 Divergence-based Faithfulness Objective

Consider a predictive model g_θ in which an intermediate calculation is later employed to justify predictions:

$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y g_\theta(x, IC(x), y) \tag{4}$$

where $IC(x)$ is the intermediate calculation on the input. A concrete example for $IC(x)$ would be the context vector calculated by the attention mechanism.

Hypothesis If there exists an intermediate calculation $IC'(x)$ that conveys a contradictory post-hoc attention compared to $IC(x)$, then $IC(x)$ cannot be regarded as faithful for predicting $\hat{y}$. If $IC(x)$ is faithful, we expect the model to *diverge* from predicting $\hat{y}$ when $IC'(x)$ is employed instead.

Based on our hypothesis, we propose a divergence-based objective which mimics behavior of a faithful explanation under stress test:

$$\mathcal{F}_{faith} = \log p(\hat{y}|x, IC'(x)) \tag{5}$$

Here $IC'(x)$ is a stress test. This objective promotes reduction in output probability under an adversarial intermediate calculation (Figure 2). It is worth noting that this objective can be potentially employed in models where outputs are modeled as soft probabilities and thus is not limited to NMT. To put model under various stress tests we manipulate the context vector during training time by changing the attention weights and feed it to the decoder to calculate the probability. More precisely:

$$\begin{aligned}\mathcal{F}_{faith} = \lambda_{zom} \log p(\hat{y}|x, IC'_{zom}(x)) \\ + \lambda_{uni} \log p(\hat{y}|x, IC'_{uni}(x)) \\ + \lambda_{perm} \log p(\hat{y}|x, IC'_{perm}(x))\end{aligned} \tag{6}$$

where IC'_{zom}, IC'_{uni} and IC'_{perm} are *ZeroOutMax*, *Uniform* and *RandomPermute* methods (see Sec. 2) to manipulate attention weights, respectively. $\lambda_{\{method\}}$ parameters regulate the contribution of each objective. We use the term $\mathcal{F}_{all}$ when all $\lambda_{\{method\}}$s in Eq. (6) are non-zero. Moreover, we use the term $\mathcal{F}_{\{method\}}$ when $\lambda_{\{method\}}$ is set to 1 and other regularization weights are zero.

4 Experimental Setup

We use the Czech-English (Cs-En) dataset from IWSLT2016and the German-English (De-En) dataset from IWSLT2014.We used Moses (Koehn et al., 2007) to tokenize the dataset. We use OpenNMT (Klein et al., 2017) as our translation framework. We employ a 2 layer LSTM-based encoder-decoder (Sutskever et al., 2014; Cho et al., 2014) model with global attention (Luong et al., 2015). We use Adam (Kingma and Ba, 2014) for training our models and we set the learning rate to 0.001. Models are trained until convergence. The baseline model is trained using Eqn. (2) and we call it $\mathcal{F}_{baseline}$. We refer to the objective as $\mathcal{F}_{all}$ when

Objective	Content Words				Function Words			
	ZOM	*Uniform*	*RandPerm*	*All*	*ZOM*	*Uniform*	*RandPerm*	*All*
$\mathcal{F}_{baseline}$	83%	90%	94%	78%	46%	48%	64%	33%
$\mathcal{F}_{zom}$	91%	93%	98%	86%	84%	87%	95%	74%
$\mathcal{F}_{uni}$	84%	98%	97%	83%	56%	98%	91%	54%
$\mathcal{F}_{perm}$	86%	95%	96%	83%	74%	97%	98%	71%
$\mathcal{F}_{all}$	**91%**	**99%**	**98%**	**89%**	**83%**	**98%**	**98%**	**82%**

Table 1: Faithfulness metric for the generated content and function words through different objectives in the **Czech-English** dataset. The columns are different tests included in the Eq.(1).

λ_{zom}, λ_{uni}, and λ_{perm} are set to 0.5, 0.375, and 0.125 respectively. λ_{faith} is set to 1.

5 Results and Discussion

5.1 Impact on faithfulness

To measure the effectiveness of the proposed objectives, we choose the best model in terms of provided faithfulness but within the 0.5 BLEU score of the maximum achieved BLEU score in the validation set. The reason is that we prefer a model that is both accurate and with faithful attention-based explanations. Table 1 and 2 show the performance of the different faithfulness objective functions when generating content words and function words across different attention manipulation methods in the Czech-English (Cs-En) and German-English (De-En) datasets respectively. Results indicate that the proposed divergence-based objective has been effective in increasing the faithfulness metric. $\mathcal{F}_{all}$ is the most effective objective for increasing faithfulness when all stress tests are included in Eq. (1). When using $\mathcal{F}_{all}$, faithfulness of attention-based explanations for content words is increased 78% to 89%, while that of the function words is from 33% to 82%(see *All* column in Table 1). The same reductions are from 76% to 89% for content works and from 32% to 86% for function words in De-En dataset. These results establish the effectiveness of our proposed objectives to increase the faithfulness metric. It is worth noting that increase in faithfulness of attention-based explanations for function words is much more than that of content words. This can be attributed to the fact the function words are mostly generated using the target-side information in the decoder (Tu et al., 2017; Moradi et al., 2019) and manipulating attention does not have much effect on generating them. However, our proposed faithfulness objective ($\mathcal{F}_{faith}$) seems to tighten the dependence of the decoder on the attention component. This results in much more increase in faithfulness for function words compared to such content words.[2]

	Objective	*ZOM*	*Uniform*	*RandPerm*	*All*
Content	$\mathcal{F}_{baseline}$	81%	90%	93%	76%
	$\mathcal{F}_{zom}$	91%	95%	98%	87%
	$\mathcal{F}_{uni}$	81%	98%	91%	80%
	$\mathcal{F}_{perm}$	85%	95%	97%	82%
	$\mathcal{F}_{all}$	**91%**	**98%**	**98%**	**89%**
Function	$\mathcal{F}_{baseline}$	45%	48%	64%	32%
	$\mathcal{F}_{zom}$	87%	95%	97%	82%
	$\mathcal{F}_{uni}$	60%	100%	95%	58%
	$\mathcal{F}_{perm}$	74%	97%	98%	72%
	$\mathcal{F}_{all}$	**87%**	**100%**	**99%**	**86%**

Table 2: Faithfulness metric for the generated content and function words through different objectives in the **German-English** dataset. The columns are different tests included in the Eq.(1).

5.2 Effect of training with single adversary on passing other stress tests

An interesting observation in Table 1 and 2 is that training with an adversary has positive effects on the model for passing stress tests from other types of adversaries. As an example, in Table 1 the column *Uniform* is the faithfulness metric when only Uniform test is employed in Eq. (1). When using this metric, we can observe that training a model with $\mathcal{F}_{perm}$ increased faithfulness from 90% to 95% for content words and from 48% to 97% for function words. We can see such effect in Table 2 as well. This observation indicates that training with each adversary can be beneficial for making model tolerant against other types of stress tests. It seems that training with each adversary strengthens the dependence of the decoder on the attention component which can be beneficial for passing other stress tests.

5.3 Regularization Effect

The model checkpoints used in Tables 1 and 2 were selected based on maximum increase in faithful-

[2]If this dependence is not desired, it is possible not to penalize function words in the faithfulness objective. However, relying on attention for generating function words can be helpful, not necessarily for interpretability but for dealing with long-range dependencies and, as a result, better translations.

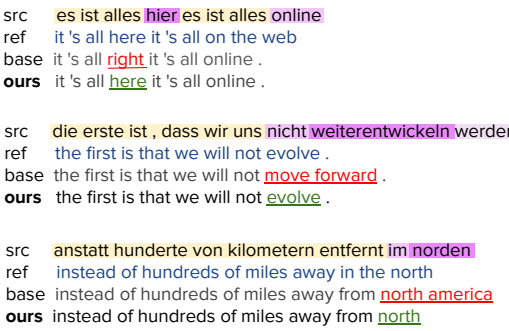


Figure 3: Some examples where our proposed objective produces better translations compared to the baseline model. In each of these cases, perturbing the attention weights has no effect on the baseline model output.

ness without sacrificing accuracy. To investigate if the proposed objective can have a general positive side effect in terms of accuracy, we train three independent models using the $\mathcal{F}_{baseline}$ and $\mathcal{F}_{all}$ objectives. Table 3 contains the average BLEU score of the trained models. It indicates that the model trained with $\mathcal{F}_{all}$, has +0.7 and +0.4 increase in BLEU score compared to the baseline for the Czech-English and German-English language pairs respectively.

	Objective	BLEU%
Cs-En	$\mathcal{F}_{baseline}$	19.68
	$\mathcal{F}_{all}$	**20.4**
De-En	$\mathcal{F}_{baseline}$	24.85
	$\mathcal{F}_{all}$	**25.21**

Table 3: BLEU score of the baseline and the model trained with $\mathcal{F}_{all}$. Pairwise bootstrap resampling (Koehn, 2004) resulted in a p-value < 0.01.

Improved BLEU scores for the faithful model can be due to two reasons: 1) the faithfulness objective can be seen as a regularization term which prevents the model from relying too much on the target-side context and the implicit language model in the decoder, which results in increased contribution of attention on the decoder and reducing some bias in the model. 2) penalizing the model for the lack of connection between justification and prediction forces the model to learn better translations by forcing it to justify each output in a *right answer for the right reason* paradigm. Figure 3 shows some examples of how our proposed model can produce better translations.

6 Related Work

While several studies have focused on understanding the semantic notions captured by attention (Ghader and Monz, 2017; Vig and Belinkov, 2019; Clark et al., 2019), evaluating attention as an interpretability approach has garnered a lot of interest. From the faithfulness perspective, (Jain and Wallace, 2019; Serrano and Smith, 2019) show that for instances in a data set there can be adversarial attention heatmaps that do not change the output of the text classifier. In other words, adversarial attention leads to no decision flip in each instance. They use this to claim that attention heatmaps are not to be trusted, or unfaithful. Wiegreffe and Pinter (2019) argue against per-instance modifications at test time for two reasons: 1) in classification tasks attention may not be useful so perturbing attention is misleading. This is not true for NMT since attention is very useful in NMT. 2) they train an adversarial attention model (e.g. uniform attention) chosen to produce attention weights distant from the original attention weights while at the same time trying to minimize classification error. They show that such adversarial attention models are not as accurate as models with attention. In our work we acknowledge that attention is useful and faithful to some extent and we aim to improve faithfulness of NMT models.

While most of these works provide evidence that attention weights are not always faithful, Moradi et al. (2019) confirm similar observations on the unfaithful nature of attention in the context of NMT models. Li et al. (2020) is one of the few papers examining attention models in NMT. However, they are focused on the task of analyzing attention weights from fidelity perspective which is different from faithfulness.

While prior works have mostly failed to explicitly distinguish faithfulness from plausibility in their arguments, Jacovi and Goldberg (2020a,b) focus on formalizing faithfulness and addressing evaluation of faithfulness separately from plausibility respectively.

7 Conclusion

In this paper, we proposed a method for quantifying faithfulness of NMT models. To optimize faithfulness we have defined a novel objective function that rewards faithful behavior through probability divergence. Unlike previous work, our method does not use prior knowledge or extraneous data. We also show that the additional constraint in the training objective for NMT does not harm translation quality and in some cases we see some better translations presumably due to the regularization effect of our faithfulness objective.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. "What is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8).

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Taipei, Taiwan. Asian Federation of Natural Language Processing.

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, Doha, Qatar. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada. Association for Computational Linguistics.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Reza Ghaeini, Xiaoli Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4952–4957, Brussels, Belgium. Association for Computational Linguistics.

Bernease Herman. 2017. The promise and peril of human evaluation for model interpretability. *arXiv preprint arXiv:1711.07414*, page 8.

Alon Jacovi and Yoav Goldberg. 2020a. Aligning faithful interpretations with their social attribution. *arXiv preprint arXiv:2006.01067*.

Alon Jacovi and Yoav Goldberg. 2020b. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion*

Volume Proceedings of the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2019. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, Copenhagen, Denmark. Association for Computational Linguistics.

Jierui Li, Lemao Liu, Huayang Li, Guanlin Li, Guoping Huang, and Shuming Shi. 2020. Evaluating explanation methods for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 365–375, Online. Association for Computational Linguistics.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Zachary C Lipton. 2018. The mythos of model interpretability. *Queue*, 16(3):31–57.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

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, Lisbon, Portugal. Association for Computational Linguistics.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2283–2288, Austin, Texas. Association for Computational Linguistics.

Sina Mohseni and Eric D Ragan. 2018. A human-grounded evaluation benchmark for local explanations of machine learning. *arXiv preprint arXiv:1801.05075*.

Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar. 2019. Interrogating the explanatory power of attention in neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 221–230, Hong Kong. Association for Computational Linguistics.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2662–2670.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

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.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. 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 *Advances in neural information processing systems*, pages 5998–6008.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. 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*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Document-level Neural Machine Translation Using BERT as Context Encoder

Zhiyu Guo
Japan Advanced Institute
of Science and Technology
guozhiyu@jaist.ac.jp

Minh Le Nguyen
Japan Advanced Institute
of Science and Technology
nguyenml@jaist.ac.jp

Abstract

Large-scale pre-trained representations such as BERT have been widely used in many natural language understanding tasks. The methods of incorporating BERT into document-level machine translation are still being explored. BERT is able to understand sentence relationships (Devlin et al., 2019) since BERT is pre-trained using the next sentence prediction task. In our work, we leverage this property to improve document-level machine translation. In our proposed model, BERT performs as a context encoder to achieve document-level contextual information, which is then integrated into both the encoder and decoder. Experiment results show that our proposed method can significantly outperform strong document-level machine translation baselines on BLEU score. Moreover, the ablation study shows our method can capture document-level context information to boost translation performance.

1 Introduction

Recent years have witnessed the great success of neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017). NMT systems have even achieved human parity on resource-rich language pairs (Hassan et al., 2018). However, standard NMT systems perform translation only at the sentence level, which ignores the dependencies among sentences when translating entire documents. To address the above challenges, various document-level NMT models, have been proposed to extract contextual information from surrounding sentences and have achieved substantial improvements in generating consistent translations (Voita et al., 2018; Zhang et al., 2018; Werlen et al., 2018; Maruf et al., 2019; Ma et al., 2020).

Large-scale pre-trained text representations like GPT-2 (Radford et al., 2018, 2019), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), have been widely used in many natural language understanding tasks. Among them, BERT is one of the most effective representations that has inspired many other representations such as RoBERTa, ALBERT. It significantly boosts the performance of many natural language understanding tasks, including text classification, question answering, etc (Devlin et al., 2019). There have been few recent attempts to incorporate BERT into NMT models (Xiong et al., 2019; Zhu et al., 2020; Weng et al., 2019; Chen et al., 2020).

Intuitively, as one of BERT's pre-training tasks is the binarized next sentence prediction (NSP) task, a natural assumption is that the NSP task has enabled the model to understand the relationship between two sentences, the relationship information is helpful to model the context information for document-level machine translation.

In this work, we propose to extend the Transformer model to take advantage of BERT document-level contextual representation. We use the pre-trained BERT as a context encoder to achieve document-level representation, which is then integrated into both the encoder and the decoder of Transformer NMT model. We use a multi-head attention mechanism and context gate to control how each layer interacts with BERT context representations adaptively.

We conducted experiments on two document-level machine translation datasets. Experimental results show that our proposed model can outperform Transformer baselines and previous state-of-the-art document-level NMT models on BLEU score. Also, we perform an ablation study showing that the BERT context encoder can capture document-level context representation to improve translation performance.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 94–100

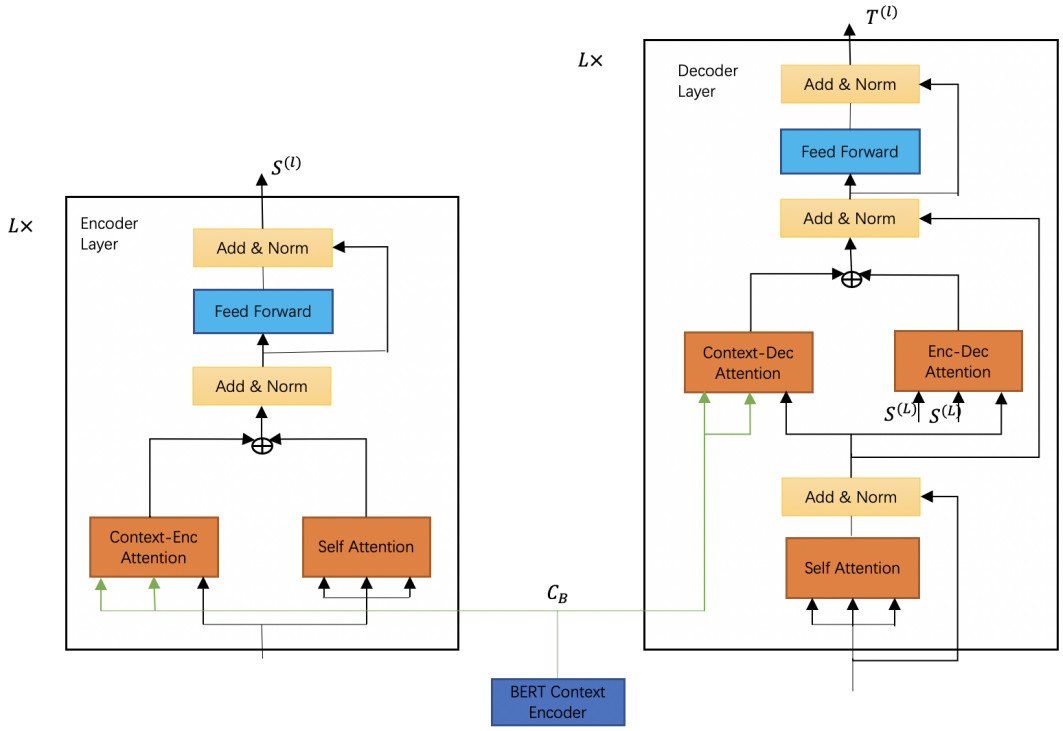


Figure 1: Illustration of using BERT as context encoder for document-level NMT model. C_B denote the output of BERT context encoder, $S^{(L)}$ denote the last layer output of Transformer encoder

2 Approach

2.1 Problem Statement

Formally, denote $\mathbf{X} = x_1, x_2, \ldots, x_N$ as a source-language document with N source sentences. The corresponding target-language document is denoted by $\mathbf{Y} = y_1, y_2, \ldots, y_N$. We assume that (x_i, y_i) is a parallel sentence pair.

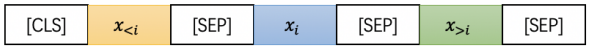
Following (Zhang et al., 2018), we omit the target-side document-level context $y_{<i}$ because of the translation error propagation problem, and source side document-level context $x_{<i}$ conveys the same information with $y_{<i}$. Therefore, the probability can be approximated as:

$$P(Y \mid X; \theta) \approx \prod_{i=1}^{N} P\left(y_i \mid x_i; x_{<i}; x_{>i}; \theta\right) \quad (1)$$

where x_i is the source sentence aligned to y_i, $x_{<i}$ and $x_{>i}$ are the document-level context sentences used to translate y_i.

2.2 BERT Context Encoder

The context encoder is a BERT model. The input x_{ctx} of BERT is the concatenation of current sentence x_i and document-level context sentences $(x_{<i}, x_{>i})$ as follows:



Where [CLS] and [SEP] are special tokens for BERT. The context input x_{ctx} is encoded by BERT into document-level context representation $\mathbf{C}_B = BERT(x)$. $\mathbf{C}_B$ is the output from the last layer of BERT.

2.3 BERT Context Representation Integration

Inspired by (Zhang et al., 2018; Zhu et al., 2020), we use multi-head attention to integrate BERT context representation $\mathbf{C}_B$ into both the encoder and the decoder of Transformer NMT model.

2.3.1 Integration into the Encoder

As shown in Figure 1, we follow Vaswani et al. (2017) using a stack of L identical layers to encode x_i. Every layer consists of two attention models with different parameters. The first attention model is a multi-head self-attention:

$$\mathbf{B}^{(l)} = MultiHead\left(\mathbf{S}^{(l-1)}, \mathbf{S}^{(l-1)}, \mathbf{S}^{(l-1)}\right) \quad (2)$$

where $\mathbf{S}^{(0)}$ denotes the word embedding of sequence x_i. The second attention model is context

attention that integrate BERT document-level context into the encoder:

$$\mathbf{D}^{(l)} = MultiHead\left(\mathbf{S}^{(l-1)}, \mathbf{C}_B, \mathbf{C}_B\right) \quad (3)$$

If we directly combine the outputs of the two attention mechanisms, the influence of document-level context will be enhanced in an uncontrolled way as the context information will be added to every layer. Also, different source sentences require different amount of context information for translation. Inspired by context gate in Werlen et al. (2018); Zhang et al. (2018), we propose to use context gate to combine the output of the two attention mechanisms.

$$\begin{aligned} g^l &= \sigma\left(W_g^l\left[\mathbf{B}^{(l)}, \mathbf{D}^{(l)}\right] + b_g^l\right) \\ \mathbf{A}^{(l)} &= g^l \odot \mathbf{B}^{(l)} + \left(1 - g^l\right) \odot \mathbf{D}^{(l)} \end{aligned} \quad (4)$$

Where σ is a sigmoid function. Then the combination is further processed by a position-wise fully connected feed-forward neural network $FFN(.)$:

$$\mathbf{S}^{(l)} = FFN(\mathbf{A}^{(l)}) \quad (5)$$

$\mathbf{S}^{(l)}$ is the representation for the source sentence x_i and its context at the l-th layer.

2.3.2 Integration into the decoder

Similar to the encoder layer, we use context gate and attention mechanism to integrate the BERT document-level context representation into standard Transformer decoder. In the l-th layer,

$$\begin{aligned} \mathbf{E}^{(l)} &= MultiHead\left(\mathbf{T}^{(l-1)}, \mathbf{T}^{(l-1)}, \mathbf{T}^{(l-1)}\right) \\ \mathbf{F}^{(l)} &= MultiHead\left(\mathbf{E}^{(l)}, \mathbf{C}_B, \mathbf{C}_B\right) \\ \mathbf{G}^{(l)} &= MultiHead\left(\mathbf{E}^{(l)}, \mathbf{S}^{(L)}, \mathbf{S}^{(L)}\right) \\ d^l &= \sigma\left(W_d^l\left[\mathbf{F}^{(l)}, \mathbf{G}^{(l)}\right] + b_d^l\right) \\ \mathbf{H}^{(l)} &= d^l \odot \mathbf{F}^{(l)} + \left(1 - d^l\right) \odot \mathbf{G}^{(l)} \\ \mathbf{T}^{(l)} &= FFN(\mathbf{H}^{(l)}) \end{aligned} \quad (6)$$

After achieving the final representations of the last decoder layer $\mathbf{T}^{(L)}$, the output probability of the current target sentence y_i are computed as:

$$\begin{aligned} & p\left(y_i \mid x_i, x_{<i}, x_{>i}\right) \\ &= \textstyle\prod_t p\left(y_{i,t} \mid y_{i,\leq t}, x_i, x_{<i}, x_{>i}\right) \\ &= \textstyle\prod_t softmax\left(E\left[y_{i,t}\right]^\top \mathbf{T}_{i,t}^L\right) \end{aligned} \quad (7)$$

Dataset	Sent No.	Doc len avg
TED	0.21M / 9K / 2.3K	121.4 / 96.4 / 98.7
News	0.24M / 2.2K / 3K	38.9 / 26.8 / 19.4

Table 1: Statistics of the train/valid/test corpora.

3 Experiments

3.1 Dataset

We use two English-German datasets as the benchmark datasets, which are TED and News. The corpora statistics are shown in Table 1.

- **TED**: This corpus is from the IWSLT 2017 MT track (Cettolo et al., 2012) aligned at the sentence level. Every TED talk is treated as a document.
- **News Commentary**: This corpus is from document-delimited News Commentary v11 [1] aligned at the sentence level.

We obtain the processed datasets from Maruf et al. (2019)[2]. We use the same train/valid/test datasets with Maruf et al. (2019), so that our results can be compared with previous work. We use the script of Moses toolkit[3] to tokenize the sentence. We use byte pair encoding (Sennrich et al., 2016) to segment all sentences with 30K merge operations. The evaluation metrics is BLEU (Papineni et al., 2002).

3.2 Implementation Details

Firstly, we train a Transformer sentence-level NMT model until convergence, then use the obtained model to initialize our proposed document-level model. The context encoder attention and context decoder attention are randomly initialized. The pre-trained BERT model is *bert_base_uncased*. When training our proposed document-level model, the parameter of the BERT encoder is not trainable. To balance the accuracy and the computation cost, we only use one previous sentence as the context.

We use the same model configuration with the setting of the Maruf et al. (2019). For the Transformer NMT model, the hidden size is 512, and the FFN layer dimension is 2048. The number of layers is 4; the number of attention head is 8. The

[1] http://www.casmacat.eu/corpus/news-commentary.html
[2] https://github.com/sameenmaruf/selective-attn
[3] https://github.com/moses-smt/mosesdecoder

Model	TED	News
HAN (Werlen et al., 2018)	24.58	25.03
SAN (Maruf et al., 2019)	24.62	24.84
QCN (Yang et al., 2019b)	25.19	22.37
Doc-Transformer (Zhang et al., 2018)	24.01	22.42
Transformer (Vaswani et al., 2017)	23.28	22.78
Flat-Transformer (Ma et al., 2020)	24.87	23.55
+BERT	**26.61**	24.52
BERT-fused (Zhu et al., 2020)	25.59	25.05
Our Reproduced Transformer	23.99	22.50
Our Proposed Model	26.23	**26.55**

Table 2: BLEU scores on the two document-level machine translation benchmarks

dropout (Srivastava et al., 2014) is 0.1 for sentence model and 0.2 for document-level model.

We use the Adam (Kingma and Ba, 2014) optimizer and the same learning rate schedule strategy as (Vaswani et al., 2017) with 4000 warmup steps. The batch size is limited to 4000 tokens. We also apply label smoothing to the cross-entropy loss, and the smoothing rate is 0.1. Our Transformer implementation is based on Fairseq (Ott et al., 2019).

3.3 Experimental results

We list the results of our experiments in Table 2, comparing six context-aware NMT models. For Document-aware Transformer (Zhang et al., 2018), Hierarchical Attention NMT (Werlen et al., 2018), Selective Attention NMT (Maruf et al., 2019) and Query-guided Capsule Network (Yang et al., 2019b), Flat-Transformer (Ma et al., 2020), using BERT to initialize the encoder of Flat-Transformer(+BERT). Most of the previous work's results are from Ma et al. (2020), except BERT-fused (Zhu et al., 2020). The result of BERT-fused (Zhu et al., 2020) is my re-implementation using the current sentence and one previous sentence as BERT input. The reproduced Transformer uses the 4-layers setting, which is the same as our proposed model.

As shown in Table 2, by leveraging BERT document-level context representation, our proposed model obtains 2.24 and 4.05 gains over our reproduced sentence-level Transformer baselines in BLEU score on TED and News datasets, respectively. Among them, our model achieves new state-of-the-art results on the News dataset, showing the superiority of exploiting BERT document-level context representation.

Integration	BLEU
none	22.50
encoder	25.65
decoder	25.55
both	26.55

Table 3: Effect of context integration. "none" means no BERT context representation is integrated, "encoder" means BERT context representation is only integrated into the encoder, "decoder" means BERT context representation is only integrated into the decoder, "both" means BERT context representation is integrated into both the encoder and the decoder.

Our model achieved significant improvement on the News dataset, but relatively smaller gains on the TED dataset and haven't achieved state-of-the-art performance. Since BERT is pre-trained on BooksCorpus and Wikipedia, and the document in News dataset is more similar to the pre-training corpus, BERT can better encode context information on News dataset.

3.4 Ablation study

Effect of Context Integration Table 3 shows the effect of integrating BERT context representation into the encoder and the decoder. We can find that integrating BERT context representation into the encoder brings more improvements, it is also beneficial to integrate representation into the decoder. The results indicate that the BERT context representation should be integrated into both encoder and decoder to achieve better performance.

Does the BERT encoder really capture the contextual information? Yes. Li et al. (2020) claims that the improvements of the multi-encoder

News	BLEU
Context	26.55
Random	25.96
Fixed	26.14

Table 4: BLEU scores using three context inputs

document-level NMT approach is not from leveraging of contextual information, instead, it is from the noise generated by the context encoder that can provide richer training signals. To investigate whether the BERT context encoder has captured contextual information, we follow the experimental setting in Li et al. (2020) presenting three types input for the BERT context encoder and make experiments on News dataset.

- *Context*: Concatenation of the previous sentence and the current sentence.
- *Random*: Concatenation of a sentence consisting of words randomly selected from the source vocabulary and the current sentence.
- *Fixed*: Concatenation of a fixed sentence and the current sentence.

As shown in Table 4, the performance of *Random* and *Fixed* decrease due to the incorrect context, which is different from the result in Li et al. (2020). This indicates that our proposed model can really capture the contextual information. Although the performance of *Random* and *Fixed* decreases, they can still outperform the standard Transformer model significantly. This is because current sentence usually plays a more important role in target sentence generation, and our proposed model can leverage the representation of current sentence obtained by BERT as extra representation.

4 Related Work

Document-level NMT Document-level NMT models incorporate the document-level contextual information to generate more consistent and coherent translations compared with sentence-level NMT models. Most of the existing document-level NMT models can be divided into two categories: Uni-encoder models (Tiedemann and Scherrer, 2017; Li et al., 2019; Ma et al., 2020) and dual-encoder models (Voita et al., 2018; Zhang et al., 2018; Werlen et al., 2018; Maruf et al., 2019; Yang et al., 2019b). Uni-encoder models (Tiedemann and Scherrer, 2017; Li et al., 2019; Ma et al., 2020) take the concatenation of contexts and source sentences as the input. Dual-encoder (Voita et al., 2018; Zhang et al., 2018; Werlen et al., 2018; Maruf et al., 2019; Yang et al., 2019b) models integrate an additional encoder to incorporate the contextual information into standard NMT models. Our proposed model can be categorised as a dual-encoder model. More recently, Li et al. (2020) indicates that in dual-encoder document-level NMT models, the BLEU score improvement is not attributed to the use of contextual information. We have shown that our model can really capture the contextual information to improve the BLEU score.

BERT for Neural Machine Translation Recently, some works tried to apply BERT into NMT. Song et al. (2019) proposed MASS pre-training, showing promising results in unsupervised NMT. Yang et al. (2019a); Weng et al. (2019); Chen et al. (2020) leverage knowledge distillation to acquire knowledge from BERT to NMT. Li et al. (2019); Ma et al. (2020) use BERT to initialize parameters of document-level NMT model encoder. BERT-fused model (Zhu et al., 2020) exploits the representation from BERT by integrating it into all layers of Transformer model. BERT-fused model can also be extended to document-level NMT, but our work is different in the modeling and experimental part. While Zhu et al. (2020) are mainly focusing on improving sentence-level machine translation performance, they proposed a drop-net trick to combine the output of BERT encoder and the standard Transformer encoder, our proposed context gate combination can better leverage document-level context information since it is more correspond to the fact that different source sentences require a different amount of context information for translation.

5 Conclusion

We have presented a method for leveraging BERT to capture contextual information for document-level neural machine translation. Experiments on two document-level machine translation tasks demonstrate the effectiveness of our approach. Besides, we have shown that our approach can really capture the context information to improve the translation performance.

For future work, we plan to compress our model into a light version to leverage more context sen-

tences. Also, we plan to do experiments on large-scale datasets and some other language pairs like Chinese-English.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268.

Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in bert for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7893–7905.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Bei Li, Hui Liu, Ziyang Wang, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and changliang Li. 2020. Does multi-encoder help? a case study on context-aware neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3512–3518, Online. Association for Computational Linguistics.

Liangyou Li, Xin Jiang, and Qun Liu. 2019. Pretrained language models for document-level neural machine translation. *arXiv preprint arXiv:1911.03110*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shuming Ma, Dongdong Zhang, and Ming Zhou. 2020. A simple and effective unified encoder for document-level machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3505–3511.

Sameen Maruf, André FT Martins, and Gholamreza Haffari. 2019. Selective attention for context-aware neural machine translation. In *Proceedings of NAACL-HLT*, pages 3092–3102.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

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.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

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.

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.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark. 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 *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274.

Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2019. Acquiring knowledge from pre-trained model to neural machine translation. *arXiv preprint arXiv:1912.01774*.

Lesly Miculicich Werlen, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954.

Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Modeling coherence for discourse neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7338–7345.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Yong Yu, Weinan Zhang, and Lei Li. 2019a. Towards making the most of bert in neural machine translation. *arXiv preprint arXiv:1908.05672*.

Zhengxin Yang, Jinchao Zhang, Fandong Meng, Shuhao Gu, Yang Feng, and Jie Zhou. 2019b. Enhancing context modeling with a query-guided capsule network for document-level translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1527–1537.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

A Review of Cross-Domain Text-to-SQL Models

Yujian Gan Matthew Purver John R. Woodward
School of Electronic Engineering and Computer Science
Queen Mary University of London
Mile End Road, London E1 4NS, UK
{y.gan,m.purver,j.woodward}@qmul.ac.uk

Abstract

WikiSQL and Spider, the large-scale cross-domain text-to-SQL datasets, have attracted much attention from the research community. The leaderboards of WikiSQL and Spider show that many researchers propose their models trying to solve the text-to-SQL problem. This paper first divides the top models in these two leaderboards into two paradigms. We then present details not mentioned in their original paper by evaluating the key components, including schema linking, pretrained word embeddings, and reasoning assistance modules. Based on the analysis of these models, we want to promote understanding of the text-to-SQL field and find out some interesting future works, for example, it is worth studying the text-to-SQL problem in an environment where it is more challenging to build schema linking and also worth studying combing the advantage of each model toward text-to-SQL.

1 Introduction

Text-to-SQL is a task to translate the natural language query (input) written by users into the SQL query (output) automatically. For example, in Table 3, we want to input the question in the table into the model to get the SQL output. Early work on text-to-SQL focused on small-scale domain-specific databases such as Restaurants, GeoQuery, ATIS, IMDB, and Yelp (Yaghmazadeh et al., 2017; Li and Jagadish, 2014; Iyer et al., 2017; Zelle and Mooney, 1996; Tang and Mooney, 2000; Popescu et al., 2003; Giordani and Moschitti, 2012). More recently, Zhong et al. (2017) proposed the first large-scale cross-domain text-to-SQL dataset, WikiSQL, which attracted much attention from the research community (Xu et al., 2017; Yu et al.; Dong and Lapata, 2018). Now, some models (He et al., 2019; Lyu et al., 2020; Anonymous, 2020) for WikiSQL have achieved over 90% execution accuracy, leading to the impression that the text-to-SQL problem has been solved. However, WikiSQL's complexity is limited: its SQL queries only cover a single *SELECT* column and aggregation, together with relatively simple selection predicates in the *WHERE* clauses, thus lacking in terms of complex SQL queries. To facilitate the study of complex SQL generation, Yu et al. (2018b) introduced Spider, a large-scale cross-domain text-to-SQL benchmark with complex SQL queries. Experiments on Spider show previous models designed for WikiSQL suffer a significant performance drop.

In this paper, we discuss the top models for the WikiSQL and Spider benchmarks. Since relatively high generation accuracy has already been achieved for the WikiSQL benchmark, and the SQL structures in Spider cover all SQL structures in WikiSQL, we focus more on models designed for Spider. This paper starts from the comparison of the overall paradigms of the models and then discusses the key modules used by most models. Overall, our contributions are as follows:

- We divide existing text-to-SQL models into two paradigms:
  1) Generate SQL structure ⇒ Fill schema
  2) Label the question ⇒ Generate SQL.
- We study that pretrained embeddings improve performance by improving schema linking and SQL structure generation.
- We evaluate the applicability and advantages of the reasoning assistance modules of previous work.
- We suggest three directions for the future.
  1) How to generate SQL if it is more challenging to build the schema linking.
  2) How to combine the different paradigms (in section 3) toward text-to-SQL.
  3) How to use graph neural networks to improve SQL structure generation.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 101–108
December 4 - 7, 2020.

Question:	What airports don't have departing or arriving flights?
SQL:	**SELECT** AirportName **FROM** Airports **WHERE** AirportCode **NOT IN** (**SELECT** SourceAirport **FROM** Flights **UNION** **SELECT** DestAirport **FROM** Flights)

Table 1: A complex nested SQL with set operator

2 WikiSQL and Spider

Differences: The most significant difference between WikiSQL and Spider is that SQL queries in Spider are more complex than in WikiSQL. Table 1 presents a complex SQL example from Spider, in which the question seems conceptually simple but involves several different pieces of database structure and SQL clause. Besides this, the Spider database contains several tables while there is only one table in the WikiSQL database. The presence of multiple tables introduces column and table name disambiguation problems to Spider, where none exist in WikiSQL. For example, suppose that the table 'student', 'course', and 'studentship' all contain a 'student id' column in a database. You would need to choose one 'student id' column from these tables when the question is 'Show the student id who choose math'. Multiple tables in Spider also cause the number of columns to be dozens of times more massive than WikiSQL, which increases the difficulty of choosing the correct column.

Similarities: Although WikiSQL and Spider are cross-domain settings, most SQL queries do not need domain knowledge during generation. The domain knowledge is a consensus that only exists in a specific field and will not be clearly stated in the question. For example, in a scenario where domain knowledge is needed, ask for 'good restaurant' can correspond to a *WHERE* condition 'star $>$ 3.5' since this domain rates the restaurants ranging from 0 to 5 stars.

Some different domain examples only replace the words related to schema item names, keeping the same sentence structure. Besides, most sentences directly use words related to schema item names instead of synonyms, which allows the model to locate the schema items by word matching. For example, in Table 2, the question rarely uses 'income' or other synonyms to replace 'salary' since the schema table word is 'salary'.

3 The Paradigms of Text-to-SQL

We only discuss two paradigms achieving relatively high performance in the text-to-SQL task, shown in Figure 1, ignoring the paradigms such as directly using a seq2seq to generate SQL. Most models follow the Paradigm One but are based on different neural network architecture, while IE-SQL (Anonymous, 2020) brings a new paradigm achieving the SOTA result in WikiSQL benchmark.

3.1 Paradigm One (Generate SQL structure $\Rightarrow$ Fill schema)

The most common text-to-SQL paradigm is to generate the SQL structure first and then fill the schema items (schema columns and tables). In WikiSQL, because the dataset only contains simple SQL, most models decompose the SQL synthesis into several independent classification sub-tasks. Each sub-task employs an independent classifier taking the entire sentence as input. For example, one classifier would be used to determine which column is the column in *SELECT* clause, and another separate classifier to determine which aggregation function is correct. These models include: SQLNet (Xu et al., 2017), TypeSQL (Yu et al.), SQLova (Hwang et al., 2019), HydraNet (Lyu et al., 2020), X-SQL (He et al., 2019), Coarse2Fine (Dong and Lapata, 2018) and others.

Among them, the SQLNet and TypeSQL models designed for WikiSQL have been transferred to Spider; however, their performance drops significantly. SyntaxSQLNet (Yu et al., 2018a) is the first model designed for Spider and, based on a similar idea, uses independent modules to predict different clauses. However its performance is less effective than some later models based on one unified grammar-based decoder modules (Guo et al., 2019; Bogin et al., 2019a).

Although these later models are based on one unified module, they also treat SQL structure generation and filling the schema items as separate processes. SQL structure generation depends on analysis of the sentence, while filling the schema items depends on the similarity between schema items and sentence tokens. For example, in Table 2, we test the top models (RAT-SQL (Wang et al., 2020), IRNET (Guo et al., 2019), and GNN (Bogin et al., 2019a)) in the Spider leaderboard, and all these models tend to generate wrong predictions of the type shown. This type of example can be found in the Spider development set where the database id is 'concert_singer'. The example shows that although based on a unified module, there is no strong interaction between generating SQL struc-

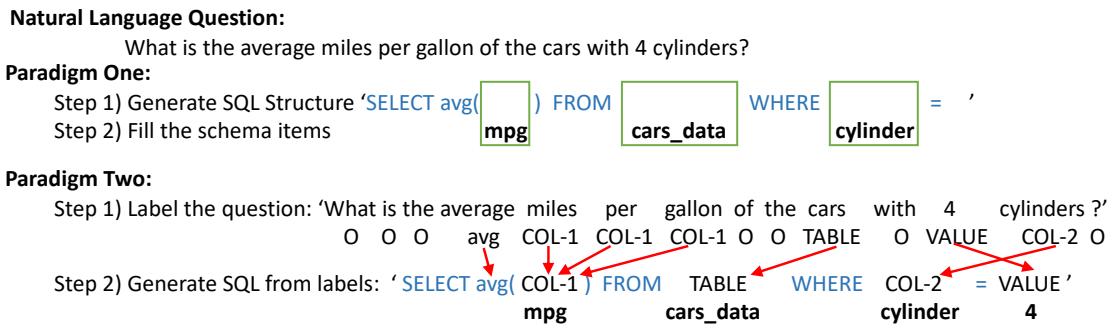


Figure 1: An example of different paradigms

Question:	What is the average salary.
Gold SQL:	**SELECT** average **FROM** salary
Wrong prediction:	**SELECT** avg(average) **FROM** salary

Table 2: A common prediction error

Question:	What are the names of houses properties?
SQL:	**SELECT** name **FROM** Properties **WHERE** type_code = 'House'

Table 3: An example of requirements for DB content

ture (generating the error 'avg' function) and filling the schema item (filling 'average' column) in the models.

3.2 Paradigm Two (Label the question ⇒ Generate SQL)

The WikiSQL SOTA model, IE-SQL (Anonymous, 2020), brings a new paradigm for text-to-SQL. At the time of writing this paper, the paper of IE-SQL is still anonymous. IE-SQL is an information extraction-based text-to-SQL method that tackles the task via sequence labeling, relation extraction, and text matching. IE-SQL first automatically labels the questions by analyzing its corresponding SQL, then trains a neural model to learn how to label a question without a SQL. Finally, IE-SQL can synthesize a SQL from the sequence labeling results in a deterministic way.

Although this approach seems to avoid the problem in Table 2, generating the correct annotation and synthesizing a SQL from the sequence labeling for Spider requires much more work than for WikiSQL because the sentences and SQL queries in the Spider are much more complicated than in the WikiSQL. The same name column that has not appeared in WikiSQL also restricts applying this method on Spider. While it is difficult to use this method on Spider directly, this method brings a new idea to solve the text-to-SQL problem. A method that combines Paradigm One and Paradigm Two is therefore worth thinking about in future.

4 Schema Linking

Schema Linking is a key module used by all the models in these two paradigms. It helps fill the schema items in Paradigm One and generate the 'COL(-∗)' and 'TABLE' label in Paradigm Two.

4.1 Schema Linking Definition

Schema linking is to establish a link between the question token and schema items. There must be a value or weight that guide a model to choose one schema item but not others. We name this value or weight as schema linking value. Any text-to-SQL model with decent performance needs a schema linking value. In Paradigm One approaches, only the schema items strongly related to the question tokens (with high schema linking value) will be filled into the SQL structure. In Paradigm Two, schema linking helps to generate the schema related labels.

4.2 Schema Linking Construction

There are different ways to construct a schema linking. The most common method is to train a neural network model that gives a higher similarity score to the link between a word token in a question and a schema item when they have the same meaning (Iyer et al., 2017). This method is widely used but may have different implementation details.

Some works implement extra schema linking by recognizing the columns and the tables mentioned in a question before training the model (Guo et al., 2019; Bogin et al., 2019a; Wang et al., 2020). It should be noted that Guo et al. (2019) and Wang

et al. (2020) name the extra schema linking as schema linking in the paper while Bogin et al. (2019a) do not mention this extra schema linking but implement it in the code. Extra schema linking is essential in these models because in the ablation study of IRNet and RAT-SQL based on Spider, removing the extra schema linking causes the biggest performance decline compared to removing other removable modules. Since the GNN paper did not present the ablation study on the extra schema linking, we conduct it in Spider and present the results in Table 4. When removing the extra schema linking from the GNN approach we observe a bigger performance decline than when removing the graph neural network component highlighted in the paper. It should be noted that some papers use the abbreviations 'GNN' or 'GNNs' refer to the graph neural networks, but in this paper, 'GNN' only represents the GNN model presented by Bogin et al. (2019a).

Besides, we can improve the extra schema linking through the database (DB) contents where the IRNet, RAT-SQL, GNN models all improve their performance by using the DB contents. For example, in Table 3, without inspecting the content of the database, it is hard to construct a linking between the word `houses` and the column '`property type code`', even by manual, since the word `houses` maybe a redundant word that often appears in questions.

However, most models in WikiSQL do not implement extra schema linking but achieve good performance. We conjecture that this is because the schema items in WikiSQL are much less than in Spider and top models use BERT (Devlin et al., 2019) to build a better schema linking than using other embedding methods. To test this conjecture, we conducted an ablation study on the GNN model and present the results in Table 4. We can see that the performance gap between GNN+BERT and GNN+BERT-ESL is smaller than between GNN and GNN-ESL. We believe BERT can compensate for part of the functionality of extra schema linking. EditSQL (Zhang et al., 2019) is another example that does not use the extra schema linking and achieves a similar improvement (around 20%) as GNN-ESL by extending BERT. It is also the highest improvement done by BERT for the models in the Spider leaderboard.

Model	Exact Match:
GNN	47.6%
GNN - ESL	24.9%
GNN - Graph Neural Networks	41.7%
GNN + BERT	49.3%
GNN + BERT - ESL	47.1%

Table 4: Ablation study results. ESL means extra schema linking.

Question:	What are the names of French singer?
SQL:	**SELECT** name **FROM** singer **WHERE** country = 'France'

Table 5: An example of requirements for pretrained word embeddings

5 Pretrained Word Embeddings

Pretrained word embeddings are also a key module widely used by most models in the two paradigms and two benchmarks. The SOTA models of the two benchmarks, belonging to different paradigms separately, are all based on BERT (Anonymous, 2020; Wang et al., 2020).

It is not surprising that BERT can improve the text-to-SQL models (Wang et al., 2020; Guo et al., 2019). As discussed in the last section, BERT provides a better embedding for schema linking than the original embeddings in the GNN model. Table 5 presents an example to illustrate why we need BERT to construct schema linking. The word '`French`' in the question cannot be constructed a schema linking through DB content since there is only '`France`' in the database. A proper pretrained word embeddings can make the distance between '`French`' and '`country`' shorter than non-pretrained embeddings.

To better understand the contribution of BERT, we list the component F1 score of RAT-SQL with and without BERT in Table 6. Spider metrics define these breakdown components according to SQL clause keywords. Among them, the '`select(no AGG)`' component represent the *SELECT* clause without aggregation function, which only include schema columns. So the F1 score of '`select(no AGG)`' depends on the accuracy of schema items appearing in *SELECT* clause. The score improvement in '`select(no AGG)`' is one more evidence that BERT can improve the schema linking. The improvement on '`keywords`' illustrate that BERT also improves the accuracy of SQL structure generation since the '`keywords`' represent

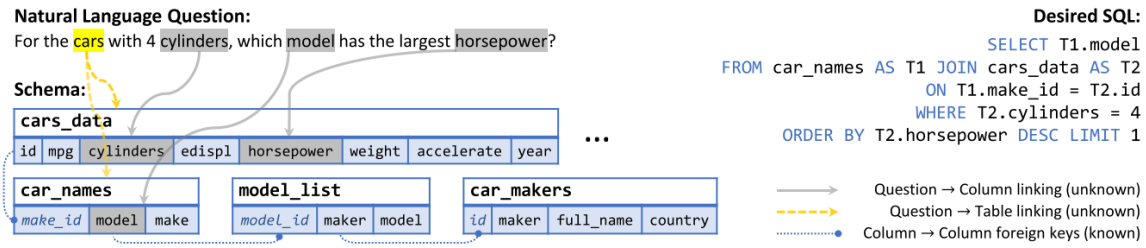


Figure 2: A challenging text-to-SQL task from the Spider dataset. (Wang et al., 2020)

the SQL structure without schema items.

Although BERT can improve the schema linking and SQL structure generation, boosting the performance by extending BERT is computational resource consuming. For example, in Table 4, we simply add the BERT to the GNN model without searching the best hyperparameter, and the performance only improves a little. The SOTA model RAT-SQL in Spider benchmark train 100 times to search the best hyperparameter for extending the BERT, which needs running about 500 days in a single TITAN RTX 24G GPU.

	RAT-SQL	RAT-SQL(BERT)
select	85.3%	90.7%
select(no AGG)	86.4%	92.1%
where	71.7%	77.1%
where(no OP)	76.0%	82.2%
group(no Having)	77.8%	83.0%
group	73.0%	80.0%
order	75.9%	83.9%
and/or	97.9%	97.7%
IUEN	47.6%	53.8%
keywords	85.3%	89.8%

Table 6: F1 scores of component matching of RAT-SQL and RAT-SQL(BERT) on dev set.

6 Reasoning Assistance Module

Some SQL clauses in Spider need reasoning to generate, but WikiSQL has almost no such clauses. For example, we cannot make out the *JOIN ON* clause directly from the question in Figure 2. Existing models mainly implement reasoning through graph neural networks and intermediate representation.

6.1 Graph Neural Networks

Graph Neural Networks in Existing Models To our knowledge, there is no WikiSQL model using graph neural networks, but some Spider models use it. The reason is that there is only one table in the WikiSQL databases. Every node that came from columns is equivalent in a graph built by only one table. Graph neural networks can not give different information or values to the equivalent nodes, which restrict the usage of graph neural networks in WikiSQL. However, if we build a graph from the table and question tokens, it may work well in WikiSQL, such as the RAT-SQL.

The models in the Spider leaderboard using graph neural networks include GNN (Bogin et al., 2019a), Global-GNN (Bogin et al., 2019b), and RAT-SQL (Wang et al., 2020). The GNN represents a schema as a graph and uses graph neural networks to embed each schema item. There are only schema item embeddings in the graph node of GNN, but the graph node in Global-GNN and RAT-SQL contains embeddings and extra schema linking data. The difference between the graph in Global-GNN and RAT-SQL is that the graph nodes in RAT-SQL include the schema items and question tokens, while nodes in Global-GNN only contain the schema items.

Benefit of Using Graph Figure 2 copied from RAT-SQL (Wang et al., 2020) illustrates the challenge of ambiguity in schema linking while '*model*' in the question refers to `car_names.model` rather than `model_list.model`. Graph neural networks can give a bigger schema linking value to the `car_names.model` than `model_list.model` by the uniquely matched `horsepower` column propagating its weight through the schema relations (e.g. foreign keys).

The other benefit of using a graph is to give the schema items that are not mentioned in the question a more significant schema linking value. Examples are often seen in *JOIN ON* clause and subqueries. In Figure 2, it is hard to construct the schema linking from question to the column `cars_data.id` and `cars_names.make_id` that appear in the *JOIN ON* clause of the SQL. The graph neural networks can construct the schema linking value for these two columns from the propagation of other linked

columns.

6.2 Intermediate Representation

Yu et al. (2018a) introduces an intermediate representation (IR) SQL that dispense with *JOIN ON* and *FROM* clauses. This IR can generate full SQL containing *JOIN ON* in a deterministic way by analyzing the schema structure. However, this IR may not generate the correct *JOIN ON* clause when there are more than one available *JOIN ON* clauses or facing the self-join. The RAT-SQL uses both graph neural networks and IR, whose IR only dispense with *JOIN ON* clauses from SQL.

Guo et al. (2019) further proposes an intermediate representation, named SemQL that removes the *GROUP BY* clause and merges the *HAVING* and *WHERE* clauses. SemQL reduces the reasoning work from SQL structure generation that does not significantly benefit from graph neural networks. However, about 20% of the generated *GROUP BY* clauses from SemQL are different from the original, which restricts its performance in Spider exact match metrics. Although most different *GROUP BY* clauses do not affect the accuracy of Spider execution match metrics, SemQL can not generate executable SQL in the current version. IR research still has room for improvement.

7 Conclusion

We discuss the existing cross-domain SOTA text-to-SQL models from the whole to the detailed modules to give a clear picture of the current text-to-SQL research progress. We illustrate that pre-trained embeddings improve the models by constructing a better schema linking and a more accurate SQL structure through experiments. This paper also provide many details that are not mentioned in the original papers, such as . However, due to space limitations, this paper cannot cover all the details of these SOTA models. We hope this paper will help you understand the key connections and differences between the previous models and have a comprehensive understanding of the text-to-SQL field.

8 Future Work

Most questions in Spider and WikiSQL directly use the words related to schema item names instead of synonyms, which means all existing models can build a schema linking by locating the same words. If you want to use these models to implement a natural language interface for database systems, you need to avoid synonyms. However, in some cases, synonyms cannot be avoided, so it is worth studying the text-to-SQL problem in an environment where it is more challenging to build schema linking.

Although only following the Paradigm Two step toward text-to-SQL in Spider needs a lot of works, a method of combining the advantages of two paradigms may boost the performance. For example, we can generate a label to every word token and then use a machine learning model to learn the word tokens with the label to generate SQL.

To improve the text-to-SQL reasoning ability, designing a new IR to simplify SQL structure generation is also a good research topic. Besides, the graph neural networks are all focused on improving the schema linking. How to use graph neural networks to improve SQL structure generation is also worth looking forward to.

Acknowledgements

We would like to thank Denis Newman-Griffis for his meticulous guidance in revising the camera-ready version and the anonymous reviewers for their helpful comments. Matthew Purver is partially supported by the EPSRC under grant EP/S033564/1, and by the European Union's Horizon 2020 programme under grant agreements 769661 (SAAM, Supporting Active Ageing through Multimodal coaching) and 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.

References

Anonymous. 2020. IE-SQL: Text-to-SQL as Information Extraction. *https://drive.google.com/file/d/1t3xEltqKpYJGYekAhQ5vYFen1ocHJ3sY/view*.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy. Association for Computational Linguistics.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global Reasoning over Database Structures for Text-to-SQL Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3659–3664, Hong Kong, China. Association for Computational Linguistics.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2012. Automatic Generation and Reranking of SQL-derived Answers to NL Questions. In *Proceedings of the Second International Conference on Trustworthy Eternal Systems via Evolving Software, Data and Knowledge*, pages 59–76.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: REINFORCE CONTEXT INTO SCHEMA REPRESENTATION. *https://www.microsoft.com/en-us/research/uploads/prod/2019/03/X_SQL-5c7db555d760f.pdf*.

Wonseok Hwang, Jinyeung Yim, and Minjoon Seo Seunghyun Park. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid Ranking Network for Text-to-SQL. *https://www.microsoft.com/en-us/research/uploads/prod/2020/03/HydraNet_20200311-5e69612887fcb.pdf*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157.

Lappoon R Tang and Raymond J Mooney. 2000. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. *Mathematics of Computation*, 22(103):651.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query Synthesis from Natural Language. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM*, pages 63:1—-63:26.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, pages 1050–1055.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*, abs/1709.0.

Multi-task Learning for Automated Essay Scoring with Sentiment Analysis

Panitan Muangkammuen[1] **and Fumiyo Fukumoto**[2]
Graduate School of Engineering[1]
Interdisciplinary Graduate School[2]
University of Yamanashi
4-3-11, Takeda, Kofu, 400-8511 Japan
{g19tk021,fukumoto}@yamanashi.ac.jp

Abstract

Automated Essay Scoring (AES) is a process that aims to alleviate the workload of graders and improve the feedback cycle in educational systems. Multi-task learning models, one of the deep learning techniques that have recently been applied to many NLP tasks, demonstrate the vast potential for AES. In this work, we present an approach for combining two tasks, sentiment analysis, and AES by utilizing multi-task learning. The model is based on a hierarchical neural network that learns to predict a holistic score at the document-level along with sentiment classes at the word-level and sentence-level. The sentiment features extracted from opinion expressions can enhance a vanilla holistic essay scoring, which mainly focuses on lexicon and text semantics. Our approach demonstrates that sentiment features are beneficial for some essay prompts, and the performance is competitive to other deep learning models on the Automated Student Assessment Prize (ASAP) benchmark. The Quadratic Weighted Kappa (QWK) is used to measure the agreement between the human grader's score and the model's prediction. Our model produces a QWK of 0.763.

1 Introduction

Automatic essay scoring (AES) is the task of grading student essays, using natural language processing to assess quality. The system is designed to reduce time and cost from the human graders' workload. Recently, neural network models based on deep learning techniques have been proposed for AES. These approaches involve the use of both recurrent neural networks, e.g., a basic recurrent unit (RNN) (Elman, 1990), gated recurrent unit (GRU) (Cho et al., 2014), or long short-term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997), and convolutional neural networks (Lecun et al., 1998; Kim, 2014). More specifically, Taghipour and Ng (2016) proposed a convolutional recurrent neural network over the word sequence to construct a document representation. Dong et al. employed hierarchical CNN and LSTM structure (Dong and Zhang, 2016; Dong et al., 2017) to construct sentences and document representation separately. Similarly, several text properties are utilized for scoring an essay, such as grammatical roles (i.e., subject, object) (Burstein et al., 2010), discourse (Song et al., 2017), or coherence (Tay et al., 2017; Mesgar and Strube, 2018).

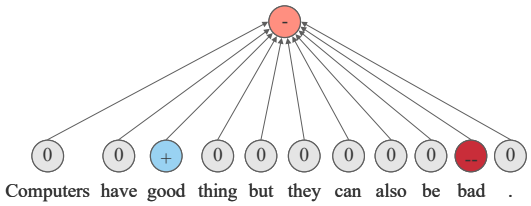


Figure 1: Example of sentiments in a sentence (This sentence is taken from an actual essay, and it is not grammatical).

The divergent and polarizing writers' opinions in their essays create overall essay structure and quality, especially in persuasive and controversial articles (Pang and Lee, 2008). Sentiment analysis has typically been designed for use with specific domains, such as movie reviews (Thongtan and Phienthrakul, 2019), product reviews (Shrestha and Nasoz, 2019), social media (Song et al., 2019), and news (Godbole et al., 2007). Beigman Klebanov et al. (2012) are the first who attempted to incorporate sentiments with essay data. It involves the use of subjective lexicons to recognize the polarity of a sentence. Another notable work (Klebanov et al., 2013) found a way to measure the compositionality of multi-word expression's sentiment profile (relative degree of polarities) in essays. Farra et al. (2015) built essay scoring systems that incorporate persuasiveness based on the analysis of opinions

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 109–116

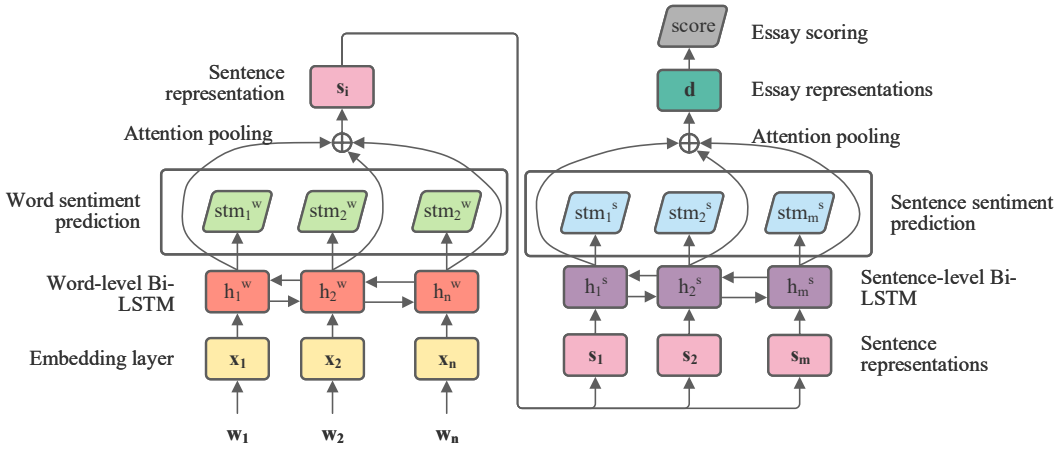


Figure 2: Multi-task Learning Framework.

expressed in the essay. Janda et al. (2019) showed sentiment-based features are in the top-ranked features giving the best performance on essay evaluation. However, these works are based on feature engineering, which must be carefully handcrafted and selected to fit the appropriate model.

Recently, Multi-Task Learning (MTL) approach has been shown promising results in many NLP tasks. The primary purpose is to leverage useful information in multiple related tasks to improve all the tasks' generalization performance (Zhang and Yang, 2017). The objectives are applied together, such as predicting the probability of the sequence and the probability that the sequence contains names (Cheng et al., 2015), the frequency of the next word with part-of-speech (POS) (Plank et al., 2016), surrounding words with other several tasks (Rei, 2017), error detection with additional linguistic information (Rei and Yannakoudakis, 2017). More advanced, Augenstein and Søgaard (2017) explored MTL for classifying keyphrase boundaries incorporating semantic super-sense tagging and identifying multi-word expression. Sanh et al. (2018) introduced a hierarchical model supervising a set of low-level tasks at the bottom layers and more complex tasks at the model's top layers. There are merely a few existing works that utilize multi-task learning in AES (Cummins et al., 2016; Cummins and Rei, 2018).

In this paper, we propose a method to incorporate sentiment analysis and AES. The proposed method utilizes the sentiment aspect for improving an essay scoring system. The sentiment information by both word-based and sentence-based, shown in Figure 1, is applied to enhance textual representations for sentiment perception of the model. To the best of our knowledge, this is the first approach on multi-task learning incorporating sentiment analysis and AES. Our proposed system is based on a hierarchical structure model to learn the features and relations between an essay score and its sentiments. The model is trained to predict a holistic score at the top-level (document-level) along with sentence and word sentiments at the lower levels, i.e., sentence-level and word-level, respectively.

2 Multi-Task Learning

We employ a hierarchical multi-task learning model similar to the model of Farag and Yannakoudakis (2019), shown in Figure 2. The model considers an essay d composed of a sequence of sentences $d = \{s_1, s_2, ..., s_m\}$, and each sentence s_i consists of a sequence of words $s_i = \{w_1, w_2, ..., w_n\}$. We describe each layer in our framework in detail.

2.1 Sentence Representation

Firstly, we consider the left-hand side of the framework in Figure 2. This part aims to learn the context representation of a sentence by taking a word sequence as an input. A word embedding lookup table maps the words in the vocabulary into low dimensional vectors,

$$\mathbf{x}_i = \mathbf{E}\mathbf{w}_i, (i = 1, 2, ..., n), \quad (1)$$

where $\mathbf{E} \in \mathbb{R}^{|V| \times D}$ be the embedding matrix, $|V|$ is the vocabulary size, and D is the word embedding dimension. $\mathbf{x}_i \in \mathbb{R}^D$ is the embedding vector of w_i, and $\mathbf{w}_i$ is a one-hot representation of w_i.

After the word embedding sequence is obtained from the embedding layer, a bidirectional LSTM is applied to the sequence to capture the context representations. In addition to a bi-direction, we concatenate the output vectors from both directions:

$$\begin{aligned}\overrightarrow{h_i^{\mathbf{w}}} &= LSTM(\mathbf{x}_i, \overrightarrow{h_{i-1}^{\mathbf{w}}}),\\ \overleftarrow{h_i^{\mathbf{w}}} &= LSTM(\mathbf{x}_i, \overleftarrow{h_{i+1}^{\mathbf{w}}}),\\ h_i^{\mathbf{w}} &= [\overrightarrow{h_i^{\mathbf{w}}}, \overleftarrow{h_i^{\mathbf{w}}}].\end{aligned} \tag{2}$$

To construct a representation of a sentence $\mathbf{s}_j$, we follow Dong et al. (2017) to use an attention pooling layer to automatically calculate weights of the word context representations obtained from the intermediate hidden states of Bi-LSTM $\{h_1^{\mathbf{w}},h_2^{\mathbf{w}},...,h_n^{\mathbf{w}}\}$:

$$\begin{aligned}u_i^{\mathbf{w}} &= tanh(W_u^{\mathbf{w}} h_i^{\mathbf{w}}),\\ a_i^{\mathbf{w}} &= \frac{exp(W_a^{\mathbf{w}} u_i^{\mathbf{w}})}{\sum_i exp(W_a^{\mathbf{w}} u_i^{\mathbf{w}})},\\ \mathbf{s}_j &= \sum_i a_i^{\mathbf{w}} h_i^{\mathbf{w}},\end{aligned} \tag{3}$$

where $W_u^{\mathbf{w}}$ and $W_a^{\mathbf{w}}$ refer to learnable parameters, $u_i^{\mathbf{w}}$ and $a_i^{\mathbf{w}}$ are the attention vector and the attention weight of the i-th word in the sentence s_j, respectively. The attention mechanism (Xu et al., 2015; Luong et al., 2015) emphasizes the salient words to build better sentence representation $\mathbf{s}_j$.

2.2 Essay Representation

An essay representation is constructed similarly to the sentence representation. Instead of taking a sequence of words $\{w_1, w_2, ..., w_n\}$ as an input, we employ another Bi-LSTM over a sequence of sentence representations $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_m\}$, as shown on the right-hand side of Figure 2:

$$\begin{aligned}\overrightarrow{h_j^{\mathbf{s}}} &= LSTM(\mathbf{x}_j, \overrightarrow{h_{j-1}^{\mathbf{s}}}),\\ \overleftarrow{h_j^{\mathbf{s}}} &= LSTM(\mathbf{x}_j, \overleftarrow{h_{j+1}^{\mathbf{s}}}),\\ h_j^{\mathbf{s}} &= [\overrightarrow{h_j^{\mathbf{s}}}, \overleftarrow{h_j^{\mathbf{s}}}].\end{aligned} \tag{4}$$

In the same way as constructing sentence representation, attention pooling is used to summarize all of the sentence contexts:

$$\begin{aligned}u_j^{\mathbf{s}} &= tanh(W_u^{\mathbf{s}} h_j^{\mathbf{s}}),\\ a_j^{\mathbf{s}} &= \frac{exp(W_a^{\mathbf{s}} u_j^{\mathbf{s}})}{\sum_j exp(W_a^{\mathbf{s}} u_j^{\mathbf{s}})},\\ \mathbf{d} &= \sum_j a_j^{\mathbf{s}} h_j^{\mathbf{s}},\end{aligned} \tag{5}$$

where $W_u^{\mathbf{s}}$ and $W_a^{\mathbf{s}}$ are learnable parameters, $u_i^{\mathbf{s}}$ and $a_i^{\mathbf{s}}$ are the attention vector and attention weight of the j-th sentence in the essay d, respectively.

2.3 Objective

Essay scoring The main task of our model is to predict the score of an essay. It is predicted by applying a fully connected layer to an essay representation $\mathbf{d}$. Then we bound the score in the range [0, 1] with a sigmoid function,

$$\hat{y} = \sigma(W^{\mathbf{d}}\mathbf{d}), \tag{6}$$

where $W^{\mathbf{d}}$ is a learnable weight matrix of a fully connected layer, and $\hat{y}$ is a predicted score. Since it is a regression task, we use mean square error (MSE) as a loss function,

$$L_{sc} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{7}$$

where N is the total number of training data, y_i is the ground-truth score, and $\hat{y}_i$ is a predicted score obtained by the model.

Sentiment Prediction Another objective of the model is to predict the sentiments of the words and sentences. To obtain such a probability distribution over word sentiment classes, we use a fully connected layer normalized by a softmax function over the hidden states of word-level Bi-LSTM $\{h_1^{\mathbf{w}},h_2^{\mathbf{w}},...,h_n^{\mathbf{w}}\}$,

$$P(stm_i^{\mathbf{w}c}|h_i^{\mathbf{w}}) = softmax(W_{stm}^{\mathbf{w}} h_i^{\mathbf{w}}), \tag{8}$$

where $W_{stm}^{\mathbf{w}}$ is a learnable weight matrix of a fully connected layer, $P(stm_i^{\mathbf{w}c}|h_i\mathbf{w})$ is a predicted probability distribution of the word i-th sentiment, and c denotes a class (e.g., positive, negative). A similar method is employed for sentences,

$$P(stm_j^{\mathbf{s}c}|h_j^{\mathbf{s}}) = softmax(W_{stm}^{\mathbf{s}} h_j^{\mathbf{s}}), \tag{9}$$

Prompts	#Essays	Genre	Avg Length	Score Range
1	1783	Persuasive / Narrative / Expository	350	2-12
2	1800	Persuasive / Narrative / Expository	350	1-6
3	1726	Source dependent responses	150	0-3
4	1772	Source dependent responses	150	0-3
5	1805	Source dependent responses	150	0-4
6	1800	Source dependent responses	150	0-4
7	1569	Persuasive / Narrative / Expository	250	0-30
8	723	Persuasive / Narrative / Expository	650	0-60

Table 1: ASAP dataset detail and statistics.

where $W^{\mathbf{s}}_{stm}$ is a learnable weight matrix of a fully connected layer, and $P(stm^{\mathbf{sc}}_j|h^{\mathbf{s}}_j)$ is a predicted probability distribution of the sentence j-th sentiment. The word and sentence sentiment prediction losses are calculated by using the negative log-probability of the correct sentiment labels,

$$L_w = -\sum_i \sum_j \sum_c stm^{\mathbf{wc}}_{ij} logP(stm^{\mathbf{wc}}_{ij}|h^{\mathbf{w}}_{ij}), \tag{10}$$

$$L_s = -\sum_j \sum_c stm^{\mathbf{sc}}_j logP(stm^{\mathbf{sc}}_j|h^{\mathbf{s}}_j), \tag{11}$$

where i indicates a number of words in a sentence, j refers to the number of sentences in an essay. c shows the number of classes of sentiment. $stm^{\mathbf{w}}_{ijc}$ and $stm^{\mathbf{s}}_{jc}$ indicate the ground-truth labels of word and sentence sentiment, respectively.

To learn in a multi-task manner, the model optimizes the total loss of the main and auxiliary objectives with different weight indicators, as shown in Eq. (12).

$$L_{total} = \alpha L_{sc} + \beta L_w + \gamma L_s, \tag{12}$$

where α, β, and $\gamma \in [0, 1]$ are hyperparameters.

3 Experiments

3.1 Dataset

In our experiments, we used the Automated Student Assessment Prize (ASAP)[1] public dataset on Kaggle to evaluate our methods. The dataset contains eight different prompts of the essay, as described in Table 1. The prompts elicit responses of different genres and of different lengths. The essays were written by students ranging from grade 7 to grade 10 and graded by at least two human graders.

We followed Taghipour and Ng (2016) to use 5-fold cross-validation for the evaluation with the same splits. In 5 folds, three folds of the data are used as a training set, one fold as the development set, and one fold as the test set. The final result is then calculated from the average of the five folds.

3.2 Sentiment annotation

We tokenize an essay into sentences and extract its sentiments using the Stanford CoreNLP[2] based on Recursive Neural Tensor Network (Socher et al., 2013). It first split an essay into sentences, then annotate each sentence and the words in it with sentiment labels, as shown in Figure 1. The extracted sentiments are represented within five sentiment classes, i.e., very negative, negative, neutral, positive, and very positive. The model was trained on the Standford Sentiment Treebank dataset extracted from movie reviews.

3.3 Evaluation Metric

The Quadratic Weighted Kappa (QWK) is used as the evaluation metric, which measures correlation or agreement between two raters (Yannakoudakis and Cummins, 2015), since it is the official evaluation metric of the ASAP competition. The QWK score ranges from 0 to 1. It can also become negative if there is less agreement than expected by chance. Therefore, the higher the value of QWK, the better the results. It is calculated using

$$K = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}, \tag{13}$$

where matrices w, O, and E are the matrices of weights, observed scores, and expected scores, respectively. A value of $O_{i,j}$ denotes the number of essays that receive a score i by the first-rater and a score j by the second-rater. The weights are

[1] https://www.kaggle.com/c/asap-aes

[2] https://nlp.stanford.edu/sentiment/

Method	Prompts								
	1	2	3	4	5	6	7	8	Avg QWK
Single-task	0.827	0.667	0.673	0.801	**0.820**	0.814	0.802	0.688	0.762
Multi-task (word sentiment)	**0.833**	0.685	0.690	0.795	0.812	**0.816**	0.798	0.673	0.763
Multi-task (sentence sentiment)	0.818	0.674	0.683	0.786	0.786	0.812	0.786	0.666	0.751
Multi-task (word&sentence sentiment)	0.803	0.658	0.664	0.772	0.799	**0.816**	0.787	0.644	0.743
Dong and Zhang (2016)	-	-	-	-	-	-	-	-	0.734
Taghipour and Ng (2016)	0.775	**0.687**	0.683	0.795	0.818	0.813	**0.805**	0.594	0.746
Dong et al. (2017)	0.822	0.682	0.672	**0.814**	0.803	0.811	0.801	**0.705**	**0.764**
Tay et al. (2017)	0.832	0.684	**0.695**	0.788	0.815	0.810	0.800	0.697	**0.764**

Table 2: Experimental results. Best result is in bold and 2nd best is underlined.

$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$, where N is the number of possible scores, matrix E is calculated as the outer product between two histogram vectors of the scores. Then matrices O and E are normalized to have the same sum and then calculate the QWK score.

3.4 Baselines

We compare our model with several baselines:

Single-task uses only one objective, which is essay scoring, without any utilization of sentiment analysis. The model is an attention-based hierarchical Bi-LSTM. The loss weight indicator α is fixed to 1, β, and γ are fixed to 0 in Eq. (1).

Multi-task has a combination objective of essay scoring and sentiment prediction. Multi-task (word sentiment) focuses only on the prediction of word sentiment. Hence, the loss weight indicator γ is fixed to 0 in Eq. (1). Similarly, the loss weight indicator β is set to 0 for multi-task (sentence sentiment) for switching off the task.

We also compare our model with several neural deep learning approaches for AES:

Hierarchical CNN (Dong and Zhang, 2016) comprises two layers of CNN, in which one convolutional layer is used to extract sentence representations, and the other is stacked on sentence vectors to learn essay representations. Concatenation of max-pooling and average pooling is used to produce the sentence and essay vectors.

RNN (Taghipour and Ng, 2016) with long short-term memory units (LSTM). LSTM units make use of three gates to forget or pass the information through time. They showed that using a mean-over-time layer is much more effective than using the last state vector or attention mechanism.

Attention-based RCNN (Dong et al., 2017) is similar to hierarchical CNN. Instead, the convolutional layer is replaced by an LSTM layer at the sentence-level to learn global coherence. Above the CNN layer and LSTM layer, an attention pooling layer is employed to acquire sentence representations and essay representations, respectively.

SKIPFLOW (Tay et al., 2017) is based on long short-term memory (LSTM) network. SKIPFLOW mechanism possesses a neural tensor layer to model the relationship between two positional outputs of LSTM across time steps. The tensor generates a coherence feature and also acts as an auxiliary memory. The coherence feature vector is then concatenated with the essay representation obtained from a mean pooling over the entire LSTM layer's hidden states.

3.5 Implementation Setup

In the embedding layer, we used the pre-trained word embedding GloVe[3] (Pennington et al., 2014) trained on 6 billion words from Wikipedia 2014 and Gigaword 5. During the training process, word embeddings are fine-tuned. The vocabulary was set to the 4,000 most frequent words by following Taghipour and Ng (2016) and treating other words as unknown words. We set the number of the essay sentences to the maximum for each essay prompts and the maximum sentence length to 128 and trained the models on batch size 16 for 50 epochs. The following hyperparameters were tuned by using optuna[4] in 100 trials.

- Embedding dimension: {50, 100, 200, 300}
- LSTM dimension: {50, 100, 200, 300}
- Optimizer: {RMSprop, Adam}
- Learning rate: [0.0001, 0.01]
- Dropout rate: [0.1, 0.5]
- α, β and γ: [0, 1]

[3]https://nlp.stanford.edu/projects/glove/
[4]https://optuna.org/

Prompts	Multi-task objective weights								
	word sentiment			sentence sentiment			word&sentence sentiment		
	α	β	γ	α	β	γ	α	β	γ
1	0.9	0.9	-	0.4	-	0.2	0.3	0.2	0.1
2	1.0	0.1	-	0.7	-	0.5	0.7	0.9	0.6
3	0.8	0.2	-	0.6	-	0.2	0.9	0.9	0.5
4	0.5	0.2	-	0.8	-	0.1	0.8	0.1	0.2
5	0.8	0.7	-	1.0	-	0.2	0.7	0.1	0.4
6	0.9	0.1	-	1.0	-	0.1	1.0	0.1	0.1
7	1.0	0.4	-	1.0	-	0.1	0.7	1.0	0.1
8	0.4	0.2	-	1.0	-	0.1	0.8	0.8	0.1

Table 3: Multi-task objective weights after tuning hyperparameters by 5-fold cross-validation. α indicates the scoring task weight, β and γ indicate the weights of the word and sentence sentiment prediction tasks, respectively

4 Results and Discussion

We can see from Table 2 that the results obtained from the Single-task already give good results compared to the existing deep learning approaches. The Multi-task (word sentiment) performed slightly better than the Single-task on four prompts, and the average result is close. The Multi-task (sentence sentiment) performed better than the Single-task only on two prompts and totally worse than Multi-task (word sentiment). The complexity of sentence sentiment might be too difficult to extract beneficial information and affect the model's sharing parameters. We tracked the accuracy of the sentence sentiment prediction during the training process. The model could reach only 70-80% accuracy. In contrast to Multi-task (word sentiment), the model could reach up to 99% accuracy of word sentiment prediction. The Multi-task (word&sentence) performed the worst among Multi-task and Single-task. The sentence sentiment prediction task's difficulty and two auxiliary tasks might make the main objective of the model, essay scoring, unstable.

Table 3 reports the Multi-task objective weights after tuning hyperparameters. We observe the model could find the proper objective weights for Multi-task (word sentiment) and Multi-task (sentence sentiment). The model performed best when the main objective weight α is greater than auxiliary objective weights, β, and γ. In contrast, in Multi-task (word&sentence sentiment), the model seems unable to find proper objective weights to balance the main and auxiliary tasks. As the summation of auxiliary task weights, β, and γ, is larger than that of α.

Comparing with other related neural deep learning models on AES, we also found that Single-task and Multi-task (word sentiment) are better than Dong and Zhang (2016) and Taghipour and Ng (2016) and comparable to Dong et al. (2017) and Tay et al. (2017). Our models perform poorly on prompt 8. One reason is that prompt 8 has the longest average length, and we limit the sentence length too short. We need to utilize the full text of an essay to the models for further improvement.

5 Conclusion

In this paper, we described a neural approach incorporating sentiment analysis and automatic essay scoring (AES). Our method is based on a hierarchical structure multi-task learning model. We compared our approach to several neural deep learning approaches (Dong and Zhang, 2016; Taghipour and Ng, 2016; Dong et al., 2017; Tay et al., 2017) on the Automated Student Assessment Prize (ASAP) benchmark. Overall, our approach is competitive with the best ones. Using word sentiment with multi-task learning, we report better results on four prompts compared to the single-task. We intend to make the model more sophisticated in future work and to apply other auxiliary tasks. We also would like to investigate the use of contextualized embeddings (e.g., BERT Devlin et al. (2019)) that shows amazing performance in many NLP tasks.

Acknowledgements

We thank Dr. Danial Beck for valuable suggestions and feedback. We are grateful to the anonymous AACL reviewers for their insightful comments and suggestions. This work was supported by the Grant-in-aid for JSPS, Grant Number 17K00299, Support Center for Advanced Telecommunications Technology Research, Foundation, and KDDI Foundation Research Grant Program.

References

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346, Vancouver, Canada. Association for Computational Linguistics.

Beata Beigman Klebanov, Jill Burstein, Nitin Madnani, Adam Faulkner, and Joel Tetreault. 2012. Building subjectivity lexicon(s) from scratch for essay data. In *Computational Linguistics and Intelligent Text Processing*, pages 591–602, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 681–684, Los Angeles, California. Association for Computational Linguistics.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746, Lisbon, Portugal. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Ronan Cummins and Marek Rei. 2018. Neural multi-task learning in automated assessment. *CoRR*, abs/1801.06830.

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–799, Berlin, Germany. Association for Computational Linguistics.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring – an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1072–1077, Austin, Texas. Association for Computational Linguistics.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

Youmna Farag and Helen Yannakoudakis. 2019. Multi-task learning for coherence modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 629–639, Florence, Italy. Association for Computational Linguistics.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74, Denver, Colorado. Association for Computational Linguistics.

Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

H. K. Janda, A. Pawar, S. Du, and V. Mago. 2019. Syntactic, semantic and sentiment analysis: The joint effect on automated essay evaluation. *IEEE Access*, 7:108486–108503.

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.

Beata Beigman Klebanov, Jill Burstein, and Nitin Madnani. 2013. Sentiment profiles of multiword expressions in test-taker essays: The case of noun-noun compounds. *ACM Trans. Speech Lang. Process.*, 10(3).

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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, Lisbon, Portugal. Association for Computational Linguistics.

Mohsen Mesgar and Michael Strube. 2018. A neural local coherence model for text quality assessment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4328–4339, Brussels, Belgium. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 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 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130, Vancouver, Canada. Association for Computational Linguistics.

Marek Rei and Helen Yannakoudakis. 2017. Auxiliary objectives for neural error detection models. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *CoRR*, abs/1811.06031.

Nishit Shrestha and Fatma Nasoz. 2019. Deep learning sentiment analysis of amazon.com reviews and ratings. *CoRR*, abs/1904.04096.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Wei Song, Dong Wang, Ruiji Fu, Lizhen Liu, Ting Liu, and Guoping Hu. 2017. Discourse mode identification in essays. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 112–122, Vancouver, Canada. Association for Computational Linguistics.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification. *CoRR*, abs/1902.09314.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891, Austin, Texas. Association for Computational Linguistics.

Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring. *CoRR*, abs/1711.04981.

Tan Thongtan and Tanasanee Phienthrakul. 2019. Sentiment classification using document embeddings trained with cosine similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 407–414, Florence, Italy. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

Helen Yannakoudakis and Ronan Cummins. 2015. Evaluating the performance of automated text scoring systems. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–223, Denver, Colorado. Association for Computational Linguistics.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *CoRR*, abs/1707.08114.

Aspect Extraction Using Coreference Resolution and Unsupervised Filtering

Deon Mai
School of Computer Science
The University of Adelaide
deon.mai@student.adelaide.edu.au

Wei Emma Zhang
School of Computer Science
The University of Adelaide
wei.e.zhang@adelaide.edu.au

Abstract

Aspect extraction is a widely researched field of natural language processing in which aspects are identified from the text as a means for information. For example, in aspect-based sentiment analysis (ABSA), aspects need to be first identified. Previous studies have introduced various approaches to increasing accuracy, although leaving room for further improvement. In a practical situation where the examined dataset is lacking labels, to fine-tune the process a novel unsupervised approach is proposed, combining a lexical rule-based approach with coreference resolution. The model increases accuracy through the recognition and removal of coreferring aspects. Experimental evaluations are performed on two benchmark datasets, demonstrating the greater performance of our approach to extracting coherent aspects through outperforming the baseline approaches.

1 Introduction

Aspect-based sentiment analysis (ABSA) is a task involving the identification of key terms (words and phrases) that refer to important parts, features, attributes, or properties of a targeted product, object or service, along with associated sentimental emotions, opinions or evaluations. What started out as a simple document-level classification task (Hu and Liu, 2004), i.e., using reviews to differentiate positive from negative, has evolved into a heavily researched field of natural language processing and information retrieval (Godbole et al., 2007). As social presence becomes more standard, the need for detecting opinions in comments or reviews becomes more present. Due to the multi-perspective opinion-oriented nature of the comments, this task will require sentence or phrase-level aspect extraction. The system must be able to locate the expressions of aspects on a sentence-level, for example in the following examples, the aspects and their associated sentiment are clear; *seaweed* and *chewy*, and *coronavirus* and *hate, terrible* respectively: "*The **seaweed** was too chewy*", and "*Hate it, the **coronavirus** is terrible*".

The existing approaches for extracting aspect are in two branches: supervised and unsupervised. Supervised approaches often formulate ATE as a token-level sequence labeling problem, achieving better accuracy than unsupervised methods in general (Li and Lam, 2017; Li et al., 2018; Zhou et al., 2019; Ma et al., 2019). However, these approaches generally require annotated data and can run into domain adaptation issues. Moreover, in reality human labelling is a time-consuming and laborious work, motivating the unsupervised approach. Topic model based approaches were proposed for this purpose (Mukherjee and Liu, 2012). These approaches model the text corpus as a mixture of opinion topics, treating the task as a problem in topic coreference resolution. This process labels aspects relating to the extracted opinion topic while dealing with coreferring aspects (Stoyanov and Cardie, 2008; Brody and Elhadad, 2010; Poria et al., 2016). Although the aspects interpreted by these models express a corpus well, they aren't coherent; individual aspects are of low quality, consisting of irrelevant or distantly-related concepts. The work in (Hu and Liu, 2004) first proposed a manually-defined rule-based approach to extract product features through observing frequent nouns and noun chunks. This approach sparked the development of numerous approaches based on frequent term mining and dependency parsing (Zhuang et al., 2006). Later, the work in (Qiu et al., 2011) proposed a unique approach to learn syntactic relations using dependency trees. Although innovative, the rule-based models heavily relied on predefined rules which only worked well when the aspect terms are confined to a small group of nouns.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 117–122

In our project, we target the issue of conducting aspect-based sentiment analysis when there is the lack of labelled data, which presents a practical challenge. To this end, as a starting point, we propose an unsupervised approach for aspect extraction on the data corpus, forming the foundation of our following works. We particularly seek an advanced rule-based approach due to its efficiency and independence from manual efforts. We first extract candidate aspects using dependency parsing and coreference resolution. A careful selection process is then applied using unsupervised techniques; inspecting the candidates for duplicate and incorrect aspects. Specifically, syntactic rules are applied on the part of speech (POS) and dependency information of a document to convert it into a candidate aspect list. This candidate list is then reduced to a final list by first applying coreference resolution, removing candidates that refer to an already existing aspect to avoid duplicity. Finally, an unsupervised filtering technique is applied on the candidates, calculating the cosine similarity of an aspect's word embedding to its neighbours and removing those that don't meet an optimal threshold. Overall, our proposed approach consists of several stages where in each the candidate list is reduced. This allows our model to overcome the small noun group restraint by first taking in a broad list of noun phrases. A clustering process is applied to complete the categorisation task to an extent.

2 Methodology

The workflow of our proposed unsupervised aspect extraction method can be broken down into four sub-processes: i) Pre-processing and text handling; ii) Noun chunk extraction via dependency parsing; iii) Candidate extraction using rules and coreference resolution; and iv) Aspect term refinement.

2.1 Pre-processing

The pre-processing performed in our approach took the form of two main tasks, applied to each entry in the data set. The first task aims to remove all *stop words* from the text. A list of extremely common English words are pre-defined, representing the stop words to be removed. The next stage of pre-processing involved converting each word to it's *lemmatized* (base) form. This is done in order to ensure words correctly match their dictionary entry in the future, specifically within the *Selecting Candidate Aspects* stage, where sentiment words need to be identified from a lexicon.

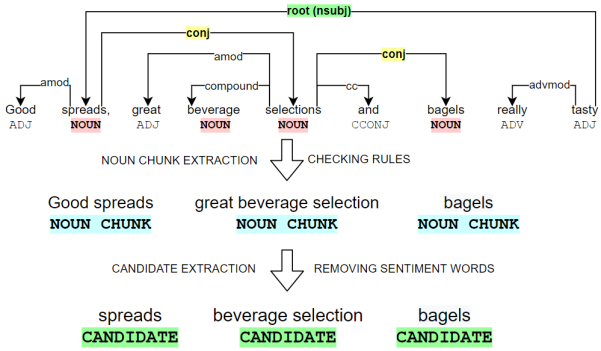


Figure 1: Example of Candidate Aspect Extraction.

2.2 Noun Chunk Extraction

For each document, the POS and dependency of each word is analysed. We particularly utilize Stanford POS tagger[1] and Stanford dependency parser[2] to get the POS tags and the dependencies in the sentences. The most important POS tags for this purpose are **NOUN** (noun), **PROPN** (proper noun) and **PRON** (pronoun). The most important relationships used are **nsubj** (nominal subject), **nsubjpass** (nominal subject, passive), **dobj** (direct object), **pobj** (preposition object), **pcomp** (preposition complement), **dative** (dative), **appos** (appositional modifier), **attr** (attribute) and **conj** (conjunct). Using these tags, all *noun chunks* are extracted from the text using a set of lexical checks. These checks are iterated through each token in the document. The first step in Figure 1 illustrates this process. Highlighted are the important POS and dependency tags, which are used to extract the noun chunks. Tokens that have an important POS tag with any of the important dependency tags *except for* **conj** are extracted as a noun chunk along with all of their syntactic descendants. The tokens with conjunct dependency are held as potential noun chunks - if the first syntactic parent of non-conjunct is found with another important dependency tag, then the original conjunct token is part of a noun chunk. Hence, it is extracted along with its syntactic descendants.

2.3 Candidate Extraction

Using the previously extracted noun chunks, a list of candidate aspects is selected using coreference

[1] https://nlp.stanford.edu/software/tagger.shtml

[2] https://nlp.stanford.edu/software/lex-parser.shtml

resolution and additional rules. Iterating through the noun chunks for each document, a step-by-step approach is taken following the processes:

2.3.1 Coreference Resolution (CoRef)

This step involves checking each noun chunk to determine if it was already mentioned previously in a different form. For example, in *"The pasta was so tasty. It also had perfect texture."*, "pasta" and "it" both corefer to the same aspect. Only the first mention of the aspect, "pasta", should be a candidate which is both "so tasty" and "perfect texture", such as:

- *Pre-CoRef*: The $\textbf{pasta}_{ASPECT}$ was so $\textbf{tasty}_{pasta}$. $\textbf{It}_{ASPECT}$ also had perfect $\textbf{texture}_{It}$
- *Post-CoRef*: The $\textbf{pasta}_{ASPECT}$ was so $\textbf{tasty}_{pasta}$. It also had perfect $\textbf{texture}_{pasta}$

We adopt (Clark and Manning, 2016) to implement the coreference resolution with minimal fine-tuning. A blacklist for resolving coreferences is created (i.e., the system will judge these words), including the following pronouns: *he, she, it, they, them, him, her, his, hers, its we*, and *us*. After performing coreference resolution, each noun chunk is tested to see if it corefers to a previously existing aspect. If so, it is removed from the candidate list.

2.3.2 Individual Token Checking

Once noun chunks satisfy the existing conditions, each token is checked individually. Observing that occasionally opinion words are included in the noun chunks, for example *"very large portions"* produces a noun chunk *"large portions"*, the token is checked to see if it is an opinion word. Using a sentiment lexicon[3], tokens are first searched for matching the surface forms, and then matching the lemma forms using their POS tags. If a corresponding match is found within the lexicon, the token is removed from the noun chunk. An example of this is illustrated in the last step of Figure 1.

2.4 Aspect Terms Refinement

Taking a similar approach to Lee et al.'s neighbourhood-based filtering technique (Lee et al., 2017), the extracted candidate aspect list is purged. The aim of this is to remove all false or incoherent aspects. Candidates that stood out had a tendency to be incorrect– if they were not closely related to other candidates, they were most likely the false aspects. Converting each a candidate into its neural word embedding form (we adopted Word2Vec as the word embedding), they are purged based on their semantic similarity to other candidates. A similarity score between two candidates is expressed as the cosine of the angle over their word vectors. The overall similarity score of a candidate is separated into two sub-scores:

- *AvgSim*: the average similarity of a candidate to all the other candidates. This is calculated by finding the sum of all the similarities and dividing the sum by the number of candidate aspects,

$$AvgSim(a) = \frac{\sum_{n \in C} similarity(a, n)}{|C|}$$

  where a is the subject candidate, *similarity(a,n)* is the similarity score between a and one candidate aspect n, and C is the list of candidate aspects.

- *MaxSim*: the maximum similarity a candidate has to another candidate.

Combining the two scores, an empirically-determined threshold is developed – candidates with similarity scores under this threshold are purged from the aspect list. If a candidate has many other similar candidates, then it coheres that the two scores will be great enough such that it will be considered a valid aspect.

3 Experiments and Results

In this section, We report our evaluations of the proposed approach on two benchmark datasets.

3.1 Experimental Setup

We introduce the datasets used to evaluate our approach, the comparative works as well as the configurations of our approach.

Datasets. We evaluated on two ***benchmark datasets***: *i) SemEval 2014 Task 4 - Restaurant* (Pontiki et al., 2014) includes 402 labelled reviews of various restaurants and cafés, used for evaluating our aspect term extraction approach against previous approaches. The dataset has been re-used in SemEval's later competitions. *ii) SentiHood* (Saeidi et al., 2016) is a labelled corpus of various urban neighbourhood discussions, in which aspects are generalised to two entities. The SentiHood dataset

[3]Bing Liu's subjectivity clues (Liu, 2010) lexicon.

Approach	Precision	Recall	$\mathbf{F_1}$
UFAL	0.50	0.72	0.59
Blinov	0.70	0.72	0.71
iTac	0.37	0.40	0.38
Pre-CoRef (ours)	0.60	**0.82**	0.70
CoRef (ours)	**0.79**	0.77	**0.78**

Table 1: Aspect Term Extraction (SemEval 2014)

focuses on categorical aspect extraction, i.e., aspect term categorization.

Comparative Approaches. In order to validate our model's performance of aspect term extraction, we compared it against three previous best-performed unsupervised approaches from the SemEval 2014 task 4 competition. We did not observe very recent works for the exact the same aspect extraction purpose. Specifically, the comparing works are *ÚFAL* (Veselovská and Tamchyna, 2014), *Blinov* (Blinov and Kotelnikov, 2014) and *iTac* (Bornebusch et al., 2014).

Configurations for Aspect Term Refinement. In purging stage, we experimented with different thresholds: the Average Similarity score threshold is tested from 10% to 40%, while the Max. Similarity threshold is tested from 50% to 75%. The best possible combination is discovered through empirical studies, presenting the most accurate purge of incorrect candidates. The *AvgSim* threshold is set to 0.2, and the *MaxSim* is set to 0.55 – candidates that are on average less than 20% similar to other words, or share less than an apex of 55% similarity to another word are purged from the candidate list.

Using our final aspect list, we plot the trained word embeddings of each aspect and perform K-Means clustering to evaluate the coherence and accuracy of aspect categories. The purpose for the clustering is because the benchmark datasets we used for evaluation include aspect category information which would help us evaluate our approach from the category perspective.

3.2 Results and Discussions

We report the evaluation results and the discussions in this section.

3.2.1 Comparisons on SemEval-14

We evaluated the performance of our model on the annotated reviews in the restaurant corpus. The criteria for assessment calculates how accurately the predictions match the true aspects. This is measured by precision, recall, and F_1 scores. The results and comparisons are summarised in Table 1.

The middle ranking of the Pre-CoRef model can be attributed to the purging process, removing incorrect aspects. Without the removal of these candidates (e.g. person relations *boyfriend, girlfriend*, and locations *New York*), the precision is considerably lower. On the other hand, the top ranking recall score, while surprising, reflects the accuracy of our rule-based system. Aspects such as person relations (e.g., *boyfriend*, *wife*) are removed, along with those not caught out by the named entity recogniser or pre-processing stage. For example, locations (e.g, *New York*) and the restaurant name itself. The change in results following the implementation of CoRef is as expected. The increased precision proves validity, as the candidate aspect list is severely reduced due to the purge of coreferring pronouns. Conversely, the reduction in recall implies meaningful aspects were also purged.

Analysing the errors, we found that if the incorrect aspect is mentioned prior to the correct aspect, the correct aspect is removed. Take the sentence "*Although it's expensive, the steak was great!*" as an example. The CoRef model identifies *it* as the original aspect, and *steak* as the coreferring aspect, hence removing *steak* from the candidate list. Our final results place our model ahead of all unsupervised approaches in all three scores.

Due to the rule-based noun chunk extraction model and similarity filtering, certain aspects were incorrectly missed. We realised that our model is incapable of identifying unconver sentiment word meanings. From "*The sweet lassi was excellent*", the correct aspect is *sweet lassi*. An error occurred here in two stage. First, our model extracts only *lassi*, as *sweet* is seen as a sentiment word and removed. During the filtering process, *lassi* is then removed as it was deemed too dissimilar to other aspects. This is most likely due to the fact that it is a foreign dish and hence a possibly uncovered word in the vocabulary.

3.2.2 Findings on SentiHood

The SentiHood dataset focuses on categorical aspect extraction. Included in the corpus are 11 categories for aspects: *live, safety, price, quiet, dining, nightlife, transit-location, touristy, shopping, green-nature*, and *multicultural*. To match these true categories, the number of clusters for the adopted K-Means algorithm is set to 11. We compared these true categories to our aspect categories obtained through grouping together our extracted aspects. In doing so, this clustering process

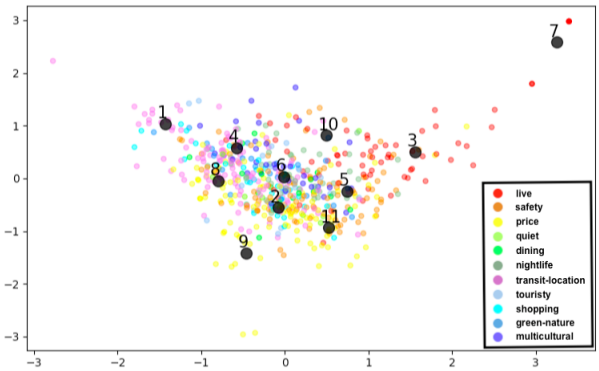


Figure 2: Clustering on Document Embeddings (SentiHood).

reveals the similarity shared between our aspect categories and the true categories; hence the coherence of our aspects. The similarity of our clusters to the true categories indicates the effectiveness of our approach from another angle. To do this, the sentences are converted into their document vector form, trained through doc2vec, and plotted. The results are seen in Figure 2. The colour of each individual point reflects the corresponding true category of that sentence. Our clusters (i.e.,, the numbers in the figure) could match the ground truth clusters. For example, cluster 3 matches the "safety" cluster. This indicates our aspect term extraction is effective. Moreover, the *live, transit-location*, and *price* categories appear to be closely related in the graph in their respective clusters. This is an extension and possible future work to our project, delving into sentiment identification and analysis.

4 Conclusions and Future Work

We have proposed and implemented an approach to aspect extraction utilising an unsupervised rule-based coreference resolution model. The basis of this approach is to apply a rule-based checking system on noun chunks extracted from the text. What started as a simple model has proven itself to be a valid approach, outperforming previous similarly unsupervised approaches. Additionally, the clusters produced on each aspect's word vector are coherent to a satisfactory level, reflecting the eligibility of our baseline model.

To improve the purging process, word vectors can be learned for a much larger vocabulary. If this can be implemented, foreign dish words such as *rasamalai* won't be incorrectly ruled out as aspects due to them not being in the vocabulary. Slang interpretations such as *rule* in "*the food options rule!*" can be investigated by using a similar technique to the stop word list. We will also involve machine learning techniques to improve the rule-based approach. Through training our model with the output of rules as an indicator feature for a discriminative learning model, we can expect that our rules are fine-tuned and adaptable to different corpora. Furthermore, to avoid mistakes in clustering where similar words included in different categories are graphed in similar locations, additional learning can be acquired by our model. Extra checks can be performed once a certain black-listed word is found in an aspect, and word embeddings can be trained further. In addition, we will perform sentiment analysis on the extracted aspects and investigate whether public sentiment can reflect the real-estate prices.

References

Pavel Blinov and Eugeny V. Kotelnikov. 2014. Blinov: Distributed Representations of Words for Aspect-Based Sentiment Analysis at SemEval 2014. In *Proc. of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 140–144, Dublin, Ireland.

Fritjof Bornebusch, Glaucia Cancino, Melanie Diepenbeck, Rolf Drechsler, Smith Djomkam, Alvine Nzeungang Fanseu, Maryam Jalali, Marc Michael, Jamal Mohsen, Max Nitze, Christina Plump, Mathias Soeken, Fred Tchambo, Toni, and Henning Ziegler. 2014. iTac: Aspect based sentiment analysis using sentiment trees and dictionaries. In *Proc. of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 351–355, Dublin, Ireland.

Samuel Brody and Noemie Elhadad. 2010. An Unsupervised Aspect-Sentiment Model for Online Reviews. In *Proc. of 2010 Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT 2010)*, pages 804–812, Los Angeles, California, USA.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2256–2262, Austin, Texas, USA.

Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *Proc. of the First International Conference on Weblogs and Social Media (ICWSM 2007)*, Boulder, Colorado, USA.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 168–177, Seattle, Washington, USA.

Annie Lee, Wenjie Zi, Afsaneh Fazly, Brandon Seibel, and Anderson De Andrade. 2017. Unsupervised Aspect Extraction from Free-form Conversations. In *Proc. of the 2017 Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM 2017)*.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018. Aspect term extraction with history attention and selective transformation. In *Proc. of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 4194–4200, Stockholm, Sweden.

Xin Li and Wai Lam. 2017. Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction. In *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2886–2892, Copenhagen, Denmark.

Bing Liu. 2010. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing, Second Edition*, pages 627–666.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring Sequence-to-Sequence Learning in Aspect Term Extraction. In *Proc. of the 57th Conference of the Association for Computational Linguistics (ACL 2019)*, pages 3538–3547, Florence, Italy.

Arjun Mukherjee and Bing Liu. 2012. Aspect Extraction through Semi-Supervised Modeling. In *Proc. of the 50th conference of the Association for Computational Linguistics (ACL 2012)*, pages 339–348, Jeju Island, Korea.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proc. of the SemEval 2014*, pages 27–35.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Federica Bisio. 2016. Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis. In *Proc. of 2016 International Joint Conference on Neural Networks (IJCNN 2016)*, pages 4465–4473, Vancouver, BC, Canada.

Baojun Qiu, Kang Zhao, Prasenjit Mitra, Dinghao Wu, Cornelia Caragea, John Yen, Greta Greer, and Kenneth Portier. 2011. Get Online Support, Feel Better – Sentiment Analysis and Dynamics in an Online Cancer Survivor Community. In *Proc. of the Third International Conference on Social Computing (SocialCom 2011)*, pages 274–281, Boston, MA, USA.

Marzieh Saeidi, Guillaume Bouchard, Maria Liakata, and Sebastian Riedel. 2016. SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods. In *Proc. of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1546–1556, Osaka, Japan.

Veselin Stoyanov and Claire Cardie. 2008. Topic Identification for Fine-Grained Opinion Analysis. In *Proc. of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 817–824, Manchester, UK.

Kateřina Veselovská and Aleš Tamchyna. 2014. ÚFAL: Using Hand-crafted Rules in Aspect Based Sentiment Analysis on Parsed Data. In *Proc. of the 8th International Workshop on Semantic Evaluation (SemEval@COLING 2014)*, pages 694–698, Dublin, Ireland.

Yan Zhou, Longtao Huang, Tao Guo, Jizhong Han, and Songlin Hu. 2019. A Span-based Joint Model for Opinion Target Extraction and Target Sentiment Classification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5485–5491, Macao, China.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie Review Mining and Summarization. In *Proc. of the 2006 ACM CIKM International Conference on Information and Knowledge Management (CIKM 2006)*, pages 43–50, Arlington, Virginia, USA.

GRUBERT: A GRU-Based Method to Fuse BERT Hidden Layers for Twitter Sentiment Analysis

Leo Horne* ETH Zurich hornel@ethz.ch
Matthias Matti* ETH Zurich mmatti@ethz.ch
Pouya Pourjafar* ETH Zurich ppouya@ethz.ch
Zuowen Wang* ETH Zurich wangzu@ethz.ch

Abstract

In this work, we introduce a GRU-based architecture called GRUBERT that learns to map the different BERT hidden layers to fused embeddings with the aim of achieving high accuracy on the Twitter sentiment analysis task. Tweets are known for their highly diverse language, and by exploiting different linguistic information present across BERT hidden layers, we can capture the full extent of this language at the embedding level. Our method can be easily adapted to other embeddings capturing different linguistic information. We show that our method outperforms well-known heuristics of using BERT (e.g. using only the last layer) and other embeddings such as ELMo. We observe potential label noise resulting from the data acquisition process and employ early stopping as well as a voting classifier to overcome it.

1 Introduction

With the rise of social media, Twitter has become an important and oft-used platform for sharing opinions and even doing politics. Furthermore, Twitter is a rich source of data collection since tweets are often closer to everyday spoken language than formally written texts. This has caused Twitter sentiment analysis (Kouloumpis et al., 2011) to become a well-established benchmark in the scientific community with practical applications such as predicting political preferences (Ansari et al., 2020). Moreover, it has piqued scientific interest in the field of natural language processing (NLP) as a source of learning informal languages. Colloquial language has many unique features as opposed to formal language, such as the use of slang words, misspellings, abbreviations, metaphors, sarcasm and context-dependent changes in meaning. Tweets also make extensive use of hashtags. This high volatility results in approaches well-suited to the analysis of formal texts giving sub-par performance on Twitter sentiment analysis.

The success of word embeddings such as Word2Vec and GloVe (Mikolov et al., 2013; Pennington et al., 2014) is based on their effectiveness in encoding semantic relationships between words. These were the first instances of word embeddings pre-trained on large corpora of text in an unsupervised fashion. However, one major drawback of these representations is that they do not account for the fact that a word can have different meanings based on its context, i.e. polysemy is not modeled. Additionally, words not in the dictionary cannot be easily taken into account in such models, which is a problem for tweets because of their frequent use of abbreviations, misspellings, and hashtags not present in the dictionary.

Following the idea of ELMo, more recent works expand unsupervised language models to a much larger scale by training them on large corpora of free text (Devlin et al., 2019; Radford, 2018; Radford et al., 2019; Brown et al., 2020). Unlike ELMo, which uses a multi-layer bi-LSTM (Hochreiter and Schmidhuber, 1997), these models are based on a multi-layer transformer architecture (Vaswani et al., 2017). Similarly to recurrent neural networks (RNNs) and LSTMs, transformers are designed to handle sequential data. However, they are entirely based on attention mechanisms (Bahdanau et al., 2014), which allow to train more powerful language models more efficiently. Transformer-based models have reached state-of-the-art performance in many NLP tasks (Liu et al., 2019b; Devlin et al., 2019; Lan et al., 2019; Radford, 2018; Radford et al., 2019; Brown et al., 2020). It has been shown that fine-tuning such pre-trained models is effective for many downstream tasks. (Howard and Ruder, 2018; Radford, 2018).

*All authors contributed equally to this work.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 123–131

A recent major breakthrough in the realm of NLP was the advent of BERT (Devlin et al., 2019), which leveraged bi-directional transformers for language representations. One of the main advantages of using BERT for Twitter sentiment analysis is that it uses sub-tokens instead of a fixed per-word token. This makes it highly suitable for the Twitter dataset that often includes misspellings and slang words. Moreover, contextualized word representations can be extracted from hidden layers of the BERT model (Devlin et al., 2019). However, one of the main challenges is the question of how and which layers to use in order to fully optimize the performance for the downstream task (Kovaleva et al., 2019; Devlin et al., 2019). Different layers of the model have been shown to capture different linguistic information (Liu et al., 2019a). While earlier layers capture more low level information such as character-based features, the middle layers tend to capture syntactic information and later layers more semantic features (Jawahar et al., 2019). Hence, finding a good way to leverage the relevant information for a specific language task becomes an important problem.

Another research problem involving microblogging is the limitations rooted in automatic data collection. Various ways of automatically labelling twitter data have been developed, such as (Pak and Paroubek, 2010; Bifet and Frank, 2010) which use emoticons to label tweets. These processes inevitably introduce label noise in the training dataset, as is also stated in (Barbosa and Feng, 2010).

Problem setup Our training dataset[1] consists of 2.5 million labeled tweets, of which one half used to contain a positive smiley ":)" and the other half a negative smiley ":(". Those which previously contained a positive smiley are labeled as positive, and those which previously contained a negative smiley are considered to be negative. Since emoticons are not a perfect indicator for positivity or negativity of a tweet (especially due to phenomena like sarcasm and irony), we suspect that the dataset potentially contains label noise.

Given a tweet, our task is to predict its sentiment as either positive or negative. We split the dataset into a 70% training dataset and a 30% validation dataset. Furthermore, we use a separate test dataset consisting of 10 thousand unlabeled tweets (excluded from the aforementioned 2.5 million tweets) to report test accuracy.

Contributions We make the following contributions:

- We propose a novel architecture to create contextualized word representations from the hidden layers of the BERT model. Our architecture learns a combination of the hidden layers using gated recurrent units (GRUs) (Cho et al., 2014).
- We show that our method outperforms well-known heuristics for this task, e.g. concatenating the last four layers or using only the last hidden layer.
- We demonstrate that our proposed method is also applicable to other BERT-based models such as RoBERTa (Liu et al., 2019b).
- Using early stopping and a voting classifier, we prevent overfitting to possible label noise (which may result from many automatic data acquisition processes) and improve generalization of the model.

2 Models and Methods

In this section, we describe our pipeline for Twitter sentiment analysis.[2]

2.1 Pre-processing

In Section 1, we mention that tweets deviate from standard written texts in that they contain many abbreviations, slang, misspellings etc. not typically found in formal written text. Since most embeddings are trained on formal texts, we pre-process the datasets in an effort to make them conform more to the type of text the embeddings were trained on. The following data pre-processing steps are performed on the training set, validation set, and test set. We delete duplicate tweets to remove biases, remove excessive whitespaces from tweets and replace <user> (resulting from Twitter @mentions) and <url> (resulting from hyperlinks) by xxuser and xxurl respectively to avoid misinterpretations due to punctuation. Moreover, we use pyspellchecker (Barros, 2018) to correct misspelled words in each tweet.

2.2 Architectures

Our architecture aims at finding a way of combining different hidden layers of BERT such that the

[1]The dataset can be accessed at https://www.kaggle.com/c/cil-text-classification-2020/data

[2]The code can be found at https://github.com/ZuowenWang0000/GRUBERT-A-GRU-Based-Method-to-Fuse-BERT-Hidden-Layers

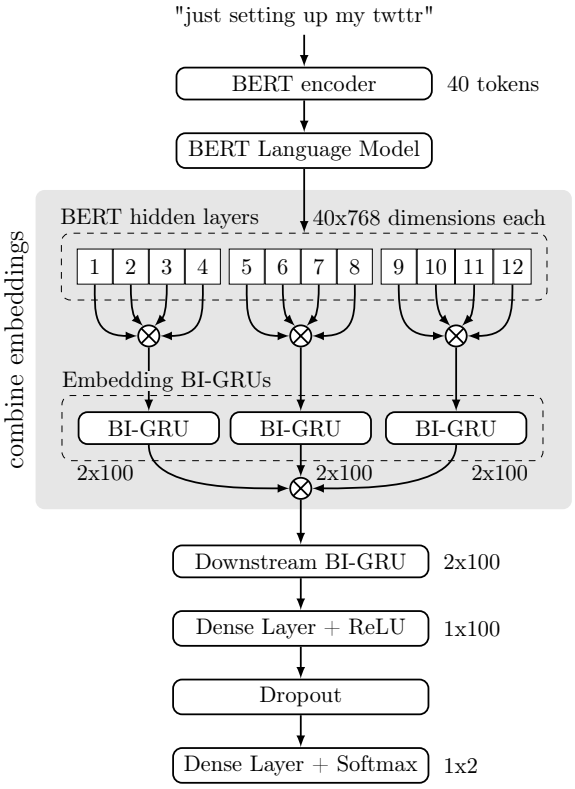


Figure 1: Illustration of the proposed architecture. The shaded part of the model indicates the combination of embeddings from hidden layers of the BERT model. The specific example shows the model *BERT-cat-3*, see Table 1. The symbol $\otimes$ represents the concatenation of tensors.

accuracy on the Twitter sentiment analysis task is increased compared to commonly used heuristics. Taking the average of the last four layers is a common heuristic which corresponds to a linear combination. Since the language used in tweets is very diverse, having a fixed way to combine the layers such as a linear combination may not leverage the full capabilities of BERT. For example, tweets including uncommon words might benefit more from information which is present in the earlier layers of BERT (character-based embeddings) whereas more formal tweets might benefit from the later layers. One possibility to overcome this challenge is to take the sequential information flow between subsequent layers into account via a recurrent unit. Hence, we opt for learning the combination of embeddings by utilizing gated recurrent units (GRUs) to capture the information flow from low level to high level features better. Moreover, we opt to first combine the BERT hidden layers in groups using a first layer of GRUs, then combine the output of the first layer of GRUs using another GRU. We hypothesize that grouping different layers together could be beneficial since the capacity of one bi-GRU could hinder its ability to capture the full information of the 12 layers. Thus, we divide the job by utilizing several bi-GRUs and assigning them grouped embeddings. We experiment with different number of bi-GRUs in order to find the best balance between incorporating information across layers and the capacity of one single bi-GRU.

An example of our model architecture is depicted in Figure 1. A tweet is first run through the BERT tokenizer, which prepares the inputs for the BERT model, i.e. tokenizes the input into sub-tokens, then embeds those sub-tokens. We heuristically clip tweets at 40 sub-tokens, since the 0.95-quantile of the number of words is 28. Shorter tweets are padded to the same length.

The tokenized tweet is then run through the BERT-base-uncased language model (Devlin et al., 2019; Wolf et al., 2019), which outputs 12 hidden layers of dimension 40×768. Each hidden layer can be interpreted as a sequence of 40 contextualized sub-token embeddings of dimension 1×768. Using a variable number of bi-GRUs, we combine multiple hidden layers into intermediate group embeddings. Each bi-GRU has a hidden state size of 100 for the forward and backward layer and creates a length 2×100-unit length embedding. We call these bi-GRUs *embedding bi-GRUs*. By concatenating the embeddings produced by embedding bi-GRUs, we obtain sub-token embeddings which contain information of all 12 layers, see the shaded area in Figure 1.

A further *downstream bi-GRU*—again with a hidden state size of 100 for both directions—is then run on the obtained embeddings. Its output is fed into a 200-unit fully-connected layer with rectified linear unit (ReLU) activation and dropout. A fully connected layer with 2 units followed by a softmax layer is added before the output is classified to either positive or negative. A cross-entropy loss is used for training the network.

Grouping hidden layers To determine the effect of group size on performance, we vary the combination of BERT hidden layers assigned to the embedding bi-GRUs. To keep the number of combinations of groups within reasonable limits, we assign the layers in uniformly sized groups to one embedding bi-GRU each, where the group size is a divisor of 12 (i.e. 1, 2, 3, 4 or 6). We further avoid shuffling the layers and only combine consecutive

layers within groups.

Model	Hidden layer groups
BERT-cat-1	1-12
BERT-cat-2	1-6, 7-12
BERT-cat-3	1-4, 5-8, 9-12
BERT-cat-4	1-3, 4-6, 7-9, 10-12
BERT-cat-6	1-2, 3-4, 5-6, 7-8, 9-10, 11-12
BERT-share-c	see *BERT-cat-c*

Table 1: Listing of different models, which differ in the number of embedding bi-GRUs used to combine hidden layers. Each group of hidden layers is assigned to one GRU. BERT-cat-c has c embedding bi-GRUs ($c \in \{1, 2, 3, 4, 6\}$) while BERT-share-c has one embedding bi-GRU shared among different groups. For *BERT-share-c* we use the same grouping as in the *BERT-cat-c* models, with the difference that the lone GRU is shared among different embedding groups. Notice that BERT-cat-1 is equivalent to BERT-share-1.

Weight sharing We observe that some of our models benefit from sharing the weights of the embedding bi-GRUs. One possible reason for this could be that different groups of BERT hidden layers contain some of the same information. Indeed, the transformer architecture has skip connections, so some information from previous layers is passed onto the next layer. We further observe that weight sharing can prevent overfitting to some extent, as it implicitly induces regularization due to the fact that the degrees of freedom are more restricted, allowing the model to be trained for more iterations.

2.3 Training and implementation details

All models are implemented with PyTorch (Paszke et al., 2019). We use pre-trained BERT models and corresponding tokenizers from huggingface's transformers (Wolf et al., 2019) library.

Dense layers are initialized with Glorot initialization (Glorot and Bengio, 2010) and a dropout rate of 0.5 is used (Srivastava et al., 2014). We use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of $1 \cdot 10^{-5}$, which is multiplied by 0.9 after each epoch. We further perform fine-tuning on the whole BERT model in every iteration in order to calibrate the embeddings with our dataset. For all experiments we use a batch size of 64 and train for 15 epochs in total. The hyperparameters are picked by a coarse grid search but due to computational resource constraint it is not exhaustive.

We train the models with one single GPU on a node equipped with an NVIDIA GeForce GTX 1080Ti and two 10-core Xeon E5-2630v4 processors. Each epoch takes approximately 1.5 hours for all models using BERT.

2.4 Preventing overfitting to label noise

Since the dataset is collected in an automated manner, there will inevitably be incorrectly labeled samples. Sarcasm and other rhetorics broadly exist in the tweets. Thus, :) and :(do not perfectly indicate the sentiment of the text. For example, *"grr .. ready for school .. i hate uniforms ! ! ugh we need our real clothes !"* is a picked sample from the training set where the label is positive but the ground truth is clearly negative.

Furthermore, the progression of the training also suggests the existence of noisy labels. In Figure 2 we show a typical training record when using a model trained exclusively on the last layer of BERT. The validation loss decreases when the validation accuracy increases at the beginning phase. However, after a turning point, the validation loss starts to rise while the validation accuracy keeps on going upwards, indicating that the classifier is less and less confident about its decisions as training progresses. This suggests that the classifier is overly considering data points with an incorrect label as opposed to correctly labeled data, making the model less confident on the latter. Notice that this phenomenon is most likely not due to a generalization problem, since the validation loss and accuracy are rising at the same time.

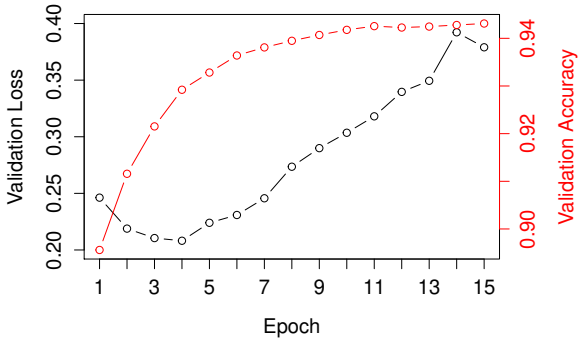


Figure 2: Validation loss and accuracy at different epochs for the model trained with the last layer of BERT with fine tuning. The validation loss reaches the lowest at the 4th epoch then starts increasing, while the validation accuracy is constantly rising until plateau. This indicates the existence of label noise in the training dataset.

In order to combat this problem, we apply early stopping and majority voting to ensure better ro-

bustness to label noise and generalization ability of the model. We assume that for the majority of the collected data samples, the sentiment corresponds to the label. Thus we aim to train the model to perform well on the data points in the test set whose labels match the actual sentiment.

Early stopping We split 30% from the preprocessed training set as a validation set, and we select the checkpoint with the lowest validation loss at the corresponding epoch, which is equivalent to early stopping on the criteria of the lowest validation loss. Li et al. (2019) suggest that overparameterized deep neural networks optimized by first-order gradient descent with early stopping are provably robust to label noise or corruption.

Majority voting Furthermore, a natural approach, which is also pointed out in (Frénay and Kabán, 2014), to alleviating this issue, and to make the model generalize better, is the use of voting classifiers. We ensemble several trained models and apply majority voting for the final prediction.

Our code framework can be easily extended to other multi-layer embeddings such as RoBERTa (Liu et al., 2019b). Therefore, we implement a voting classifier with BERT-base-uncased, RoBERTa and a multilingual BERT due to the presence of multilingual tweets in the dataset.

3 Results

In this section we first discuss the baselines we compare our models to and report in Tables 2 and 3 mean accuracies of three runs for each experiment, as well as the standard deviation. We report the accuracies of the method based on the test split of the dataset.

Major Baselines To assess the usefulness of the learned contextualized representations of our models, we implement a baseline architecture similar to the GRUBERT architecture as follows: after the embedding layer, there is a single bi-GRU (analogous to the embedding bi-GRUs from GRUBERT), followed by another bi-GRU (analogous to the downstream bi-GRU from GRUBERT), followed by a dense layer with ReLU activation, dropout, and dense layer with a softmax over the logits as in the GRUBERT architecture.

We also chose to compare the intermediate embeddings of our models with two BERT hidden layer combination heuristics used in (Devlin et al., 2019).

- *GloVe*: We use GloVe trained on Wikipedia 2014 and Gigaword 5 from Pennington et al. (2014).
- *ELMo*: We use ELMo embeddings (Peters et al., 2018) provided by the Flair NLP library (Akbik et al., 2019) (using AllenNLP (Gardner et al., 2017)).
- *BERT-last-layer*: This baseline uses the last hidden layer, i.e. layer 12.
- *BERT-last-four*: This baseline uses the concatenation of the last four hidden layers.

All baseline models are trained using the same hyperparameters as in Section 2.3, except for ELMo, which is trained with a learning rate of $1 \cdot 10^{-3}$. All baselines use one embedding bi-GRU followed by the rest of the downstream architecture, i.e. one embedding bi-GRU and a classifier.

BERT-cat and BERT-share Table 2 presents our results for the Twitter sentiment classification task using the models mentioned in Section 2. The *BERT-cat-2* and *BERT-cat-4* models outperform the equivalent parameter-sharing models, although *BERT-share-3* outperforms *BERT-cat-3*. It remains an open question why this is the case, although we suspect that it may be due to consecutive groups of four layers containing similar information to each other, while other consecutive groupings diverge more in the type of information contained in each grouping. Furthermore, we suspect that BERT-cat-2 outperforms the other BERT-cat models because BERT-cat models send different groups of layers through different bi-GRUs, thereby cutting the flow of information between layers. BERT-cat-2 only has two groups, so this cutting of information flow between layers is minimal. This hypothesis also explains why accuracy shows a downward trend as the number of groups in BERT-cat models is increased.

We also observe that certain configurations of GRUBERT outperform other commonly used embeddings such as GloVe, ELMo, as well as other common ways of using BERT embeddings, such as using only the last layer or concatenating the last four layers.

Table 3 validates our idea of using a GRU to capture the fact that different tweets may each benefit from different BERT layers by replacing the first layer of GRUs with fully connected linear layers. A linear layer always combines layers in the same way, so different tweets are always associated with the same combination of the embedding

Model	Mean Accuracy (Std. Dev.)
GloVe	83.52
ELMo	86.44
BERT-last-layer	89.06 (0.05)
BERT-last-four	89.27 (0.15)
BERT-cat-2	**89.43 (0.02)**
BERT-cat-3	89.21 (0.13)
BERT-cat-4	89.22 (0.26)
BERT-cat-6	89.05 (0.21)
BERT-share-1	89.37 (0.19)
BERT-share-2	89.04 (0.20)
BERT-share-3	**89.66 (0.18)**
BERT-share-4	89.14 (0.35)
BERT-share-6	89.02 (0.24)

Table 2: Experiment results for baselines, BERT-cat models and BERT-share models. The mean accuracy is computed over several runs of the model and evaluated on the test set. Note that *BERT-cat-1* is equivalent to *BERT-share-1*, hence it is omitted from the table. The GloVe and ELMo baselines are presented without standard deviation due to computational resource constraints and since they are significantly worse than BERT-based approaches.

layers, as opposed to a GRU, which can generate different combinations of layers for different tweets due to the recurrent information flow, and is therefore more context-sensitive. Table 3 shows that we obtain a higher accuracy using GRUs.

Model	Mean Acc. (Std. Dev.)
BERT-share-3-linear	89.43 (0.17)
BERT-share-3	**89.66 (0.18)**

Table 3: Comparison between linear layers for layer group combining vs GRUs. *BERT-share-3-linear* is equivalent to *BERT-share-3*, but with the first layer of GRUs replaced by fully connected linear layers.

Our method can be easily extended to other multi-layer embeddings such as RoBERTa (Liu et al., 2019b), leaving possibilities of adapting ensemble methods. We evaluate the final model by implementing our technique on top of a RoBERTa (Liu et al., 2019b) model and doing an ensemble with various BERT-share-3 models trained: (1) as described in Section 2, (2) on the full dataset, (3) using multilingual BERT embeddings, (4) with a weight decay of $1 \cdot 10^{-5}$, (5) using RoBERTa embeddings, as well as (6) a BERT-share-4 and (7) a BERT-share-6 model, both trained with RoBERTa embeddings. Using this technique we reach a final test score of 90.94%.

4 Discussion

We show empirically that GRUBERT is superior to standard embeddings for the task of Twitter sentiment analysis. However, our model is not easily interpretable and does not allow deeper insights into the BERT hidden layers, as is made apparent by the fact that we cannot draw concrete conclusion about why weight sharing gives a boost to certain groupings but not to others. In future work we would like to find interpretable combinations of the BERT layers for different task, to better understand the linguistic features present in each hidden layer.

As another item of future work, the effect of each component on the overall architecture should be examined more closely.

Furthermore, the effectiveness of our approach for other NLP tasks remains to be tested. However, our architecture can easily be used as a plug-in module for other multi-layer embeddings and downstream models, allowing the effectiveness to be easily examined by future work.

5 Conclusion

We have shown that a dynamic way of combining the BERT hidden layers using GRUs can lead to performance benefits in the case of irregular and plastic language found in tweets. We further experimented with different ways of combining the embeddings and observed that weight sharing can benefit the training process by implicitly inducing regularization and restricting the model complexity. We used early stopping as well as voting classifiers to address the label noise problem inherent in the automatic data collection process Using these findings, we develop a framework for the problem of machine-labeled Twitter sentiment analysis which makes use of an ensemble of different GRUBERT models to combat label noise.

Acknowledgements

We thank the Data Analytics Lab at ETH Zurich for providing computing infrastructure. We also thank them, in addition to our mentor Shuhei Kurita and the anonymous reviewers, for valuable feedback.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

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, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mohd Zeeshan Ansari, Areesha Fatima Siddiqui, and Mohammad Anas. 2020. Inferring political preferences from twitter.

Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 36–44.

Tyler Barros. 2018. pyspellchecker. Available online: https://github.com/barrust/pyspellchecker. Consulted 2020-07-27.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in twitter streaming data. pages 1–15.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

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. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Benoît Frénay and Ata Kabán. 2014. A comprehensive introduction to label noise. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*.

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.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

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*. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

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*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the 2011 International Conference on Web and Social Media (ICWSM)*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2019. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, Workshop Track Proceedings*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'é Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

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)*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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(56):1929–1958.

Shikhar Vashishth, Prateek Yadav, Manik Bhandari, Piyush Rai, Chiranjib Bhattacharyya, and Partha P. Talukdar. 2018. Graph convolutional networks based word embeddings. *CoRR*, abs/1809.04283.

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*, volume 30, pages 5998–6008. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

A Appendices

A.1 Number of Words per Tweet

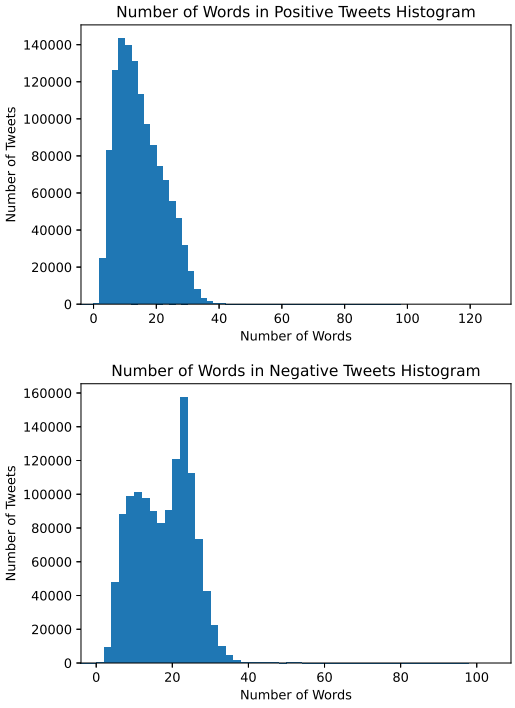


Figure 3

Figure 3 illustrates histograms of the number of words in the original data sets with positive and negative tweets.

Dataset	Positive Tweets	Negative Tweets
#Tweets	1.25mio	1.25mio
Mean	14.34	17.14
Std. Dev.	7.18	7.31
Max	128	104
Min	1	1
Quantiles		
0.05	5	6
0.25	5	6
0.5	13	18
0.75	19	23
0.95	28	28

Table 4: Statistics about number of words in tweets.

A.2 Additional Baselines

In this section, we present additional baselines using context-sensitive word embeddings such as ELMo (Peters et al., 2018), Flair (Akbik et al., 2018) and a graph convolution network based embedding SynGCN (Vashishth et al., 2018). We also try different stackings of these embeddings then feed them into the embedding bi-GRU followed by the downstream architecture. The training schedule is the same as mentioned in section 2.3. The following embeddings are used (Suffix -ft indicates with fine-tuning):

- *SynGCN*: We the pretrained SynGCN embedding from (Vashishth et al., 2018).
- *GloVe-SynGCN*: We stack the GloVe embedding used in 2 with the SynGCN embedding.
- *ELMo-mix*: On top of GloVe-SynGCN, we stack the ELMo embedding same as used in table 2. We use starting learning rate $1 \cdot 10^{-4}$ (choosed by grid search) with decay by multiplying 0.9 after every epoch.
- *Flair-mix*: On top of GloVe-SynGCN, we stack the Flair embedding same as used in table 2. For *Flair-mix-ft* fine-tuning was used.

Model	Accuracy
SynGCN	83.48
GloVe-SynGCN	85.44
ELMo-mix	86.30
Flair-mix	86.44
Flair-mix-ft	87.16

Table 5: Results from additional baselines.

Exploring Statistical and Neural Models for Noun Ellipsis Detection and Resolution in English

Payal Khullar
International Institute of Information Technology Hyderabad
Gachibowli, Hyderabad
payal.khullar@research.iiit.ac.in

Abstract

Computational approaches to noun ellipsis resolution has been sparse, with only a naive rule-based approach that uses syntactic feature constraints for marking noun ellipsis licensors and selecting their antecedents. In this paper, we further the ellipsis research by exploring several statistical and neural models for both the subtasks involved in the ellipsis resolution process and addressing the representation and contribution of manual features proposed in previous research. Using the best performing models, we build an end-to-end supervised Machine Learning (ML) framework for this task that improves the existing F1 score by 16.55% for the detection and 14.97% for the resolution subtask. Our experiments demonstrate robust scores through pretrained BERT (Bidirectional Encoder Representations from Transformers) embeddings for word representation, and more so the importance of manual features– once again highlighting the syntactic and semantic characteristics of the ellipsis phenomenon. For the classification decision, we notice that a simple Multilayar Perceptron (MLP) works well for the detection of ellipsis; however, Recurrent Neural Networks (RNN) are a better choice for the much harder resolution step.

1 Introduction

Noun ellipsis is a linguistic phenomenon where the head noun of a noun phrase gets deleted, without making the sentence ungrammatical. For example in the sentence in (1) from (Lobeck, 1995), the noun *presentation* is elided at "[e]".

1. John's **presentation** on urban development was virtually ignored because [NP Mary's [e]] was so much more interesting.

The elided information can be retrieved from the previous context as in (1) or with the knowledge of idiomatic usage of language as in *I will be back in two [e].* where *two* means *two minutes*. It is also possible that the reference of the elided information comes from extra-linguistic, situational context. For example, consider a speaker pointing towards the roses in a shop and saying an utterance as in *I will take two [e].* While human interlocutors resolve any such elided information by disambiguating from context, cognitive commonsense extension and reasoning (Chen, 2016), ellipsis resolution can be a hard task for Natural Language Processing (NLP) systems (Hardt, 1999). Resolution of ellipsis comprises two tasks - detection of the elided material and antecedent selection (Liu et al., 2016b; Nielsen, 2003). Ellipses occur in the environment of certain syntactical structures or trigger words, also known as licensors or triggers of ellipses. They are useful syntactic cues for the detection of noun ellipsis. See Figure 1 for an example of the noun ellipsis resolution process.

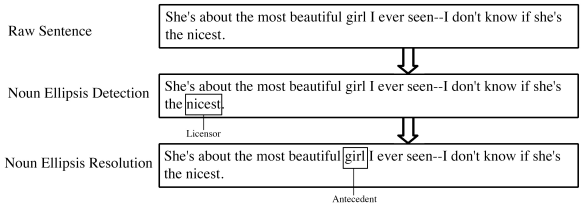


Figure 1: An example of the noun ellipsis resolution process from the dialogue L131377 of the m_44 movie of the Cornell Movie Dialogs dataset. Here, *the nicest* and *girl* denote the ellipsis licensor and antecedent respectively.

2 Related Work

Nominal ellipsis has been a topic of interest in theoretical linguistics for a very long time (Halliday and Hasan, 1976; Dalrymple et al., 1991; Lobeck, 1995; Lappin, 1996; Hobbs and Kehler, 1997; Hardt, 1999; Johnson, 2001; Wijnen et al., 2003; Merchant, 2004; Frazier, 2008; Chung et al., 2010; Mer-

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 132–138

chant, 2010; Goksun et al., 2010; Gunther, 2011; Rouveret, 2012; Lindenbergh et al., 2015; van Craenenbroeck and Merchant, 2013; Park, 2017; Hyams et al., 2017; Kim et al., 2019). Computational approaches to the ellipsis phenomenon majorly focus on the Verb Phrase Ellipsis (VPE) along with a few related phenomenon such as gapping, sluicing and do-so anaphora, for instance, the detection of VPE in the Penn Treebank using pattern match (Hardt, 1992), a transformation learning-based approach to generated patterns for VPE resolution (Hardt, 1998), the domain independent VPE detection and resolution using machine learning (Nielsen, 2003), automatically parsed text (Nielsen, 2004b), sentence trimming methods (McShane et al., 2015), linguistic principles (McShane and Babkin, 2016), improved parsing techniques that encode elided material dependencies for reconstruction of sentences containing gapping (Schuster et al., 2018), discriminative and margin infused algorithms (Dean et al., 2016), Multilayer Perceptrons (MLP) and Transformers (Zhang et al., 2019). In recent times, there has been a surge in the computational research on nominal ellipsis and closely related phenomena (Khullar et al., 2020, 2019; Lapshinova-Koltunski et al., 2018; Menzel, 2017; Menzel and Lapshinova-Koltunski, 2014). For the resolution process, we previously proposed a rule based system (Khullar et al., 2019) that detects noun ellipsis using syntactic constraints on licensors of ellipsis and resolves them by matching Part-of-Speech (POS) tag similarity between the licensor of ellipsis and the modifier of the antecedent. It later fine tunes these syntactic rules on a small curated dataset that contains 234 instances of noun ellipsis along with some negative samples (Khullar et al., 2019). For the present paper, we further the research on noun ellipses by using the NoEl corpus annotated by us previously (Khullar et al., 2020) to experiment with state-of-the-art ML models.

3 The Proposed Approach

Following the VPE resolution framework presented by (Zhang et al., 2019), we investigate a similar framework for noun ellipsis resolution in English and present alternative choices of the models at each step as shown in Figure 2. We use the NoEl corpus (Khullar et al., 2020) that marks noun ellipsis instances as a separate layer (using the stand-off annotation scheme) on the Cornell Movie Dialogs corpus (Danescu-Niculescu-Mizil and Lee, 2011). The corpus marks a total of 946 annotations, of which 438 are described as endophoric, i.e. with a textual antecedent, and 508 exophoric, i.e. without a textual antecedent.

3.1 Noun Ellipsis Detection

From a given sentence, we first select all words belonging to the syntactic categories that can license noun ellipsis in English, i.e. cardinal and ordinal numbers, determiners and adjectives (Ross, 1967; Lobeck, 1995; Mitkov, 1999; Saito et al., 2008; Kim et al., 2019; Khullar et al., 2019) using a POS tag filter. The POS tags are obtained from state-of-the-art spaCy parser (Honnibal and Johnson, 2015). For simplicity, we refer to words with these categories as noun modifiers (although in strict linguistic terms, this might be problematic). For each of these selected noun modifiers, we follow the task specification for VPE detection used by (Nielsen, 2004a; Bos and Spenader, 2011; Liu et al., 2016a; Dean et al., 2016) and present noun ellipsis detection as a binary classification task, where given a noun modifier and the sentence in which it occurs as the input, the goal of the classifier is to predict whether the noun modifier licenses a noun ellipsis or not. Formally, for a given licensor word l_i is a licensor in a sentence s, the task is represented as follows:

$$f(l_i, s) \rightarrow \{0, 1\}$$

where 1 denotes that l_i is a licensor in s, and 0 otherwise.

We experiment with both static and contextualised word embeddings for word and context representation. For the former, we choose pretrained fastText (FT) word embeddings (Bojanowski et al., 2016) as they provide representations for rare and unknown words that might be frequent in the movie dialogues. For the latter, we use pretrained BERT embeddings from the BERT base uncased word-piece model for English (Devlin et al., 2019), as these currently offer the most powerful embeddings taking into account a large left and right context.

fastText We take pretrained FT word embeddings for the noun modifier and sentence in which it is present and sum pool to obtain a single vector that we use to train our classifiers. For the statistical models, we choose Naive Bayes and Linear Support Vector Machine (SVM), and use scikit learn (Pedregosa et al., 2011) with 5-fold cross validation for training and testing. We choose a

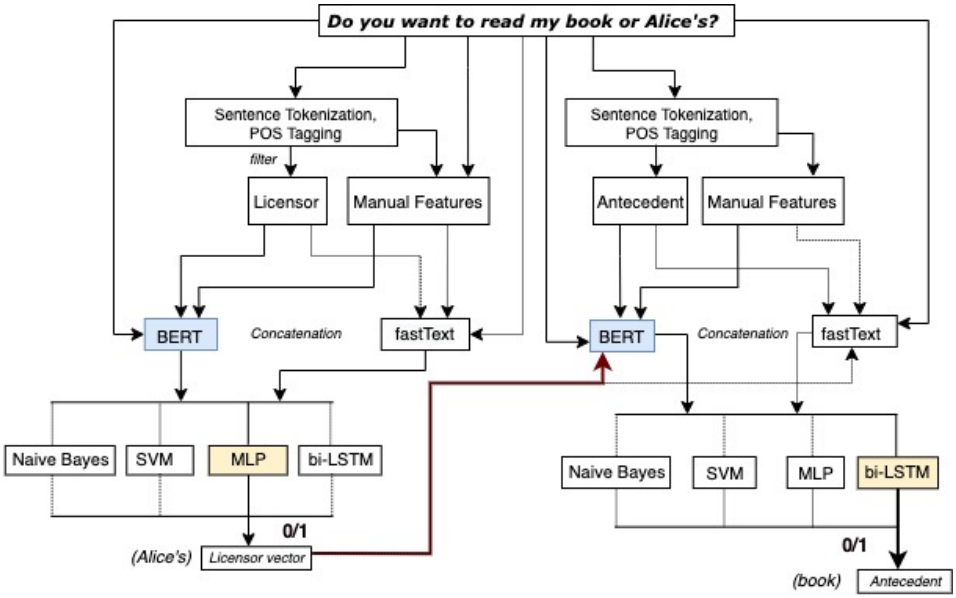


Figure 2: The proposed framework for noun ellipsis detection and resolution in English.

simple MLP and a biLSTM (Bidirectional Long Short Term Memory) as our state-of-the-art neural classifiers.

BERT We separate the sentence and the licensor with a [SEP] token and keep the sequence length to 300 as this is the maximum sentence length in the training data. After creating the concatenated set of tokens, if the number of tokens are greater than 300, we clip it to 300, otherwise we add [PAD] tokens which correspond to the embedding of 768 dimensional zero-vector. The [CLS] output of the BERT model (Devlin et al., 2019) is then fed into Naive Bayes, Linear SVM, MLP and bi-LSTM networks as above.

Manual Syntactic Features For each of these models, we additionally experiment with manual syntactic features. We use the lexical features proposed by (Dean et al., 2016) and extended lexical features by (Zhang et al., 2019), and take the five syntactic constraints on licensors of ellipsis explored by (Khullar et al., 2019) for their rule-based approach as our slot pattern features. We concatenate all these features to the embeddings from the previous step and check if they improve the classification decision.

3.2 Noun Ellipsis Resolution

We define noun ellipsis resolution as a binary classification task where given a licensor, antecedent candidate and their context, the goal of the classifier is to predict whether the antecedent candidate is the resolution of the ellipsis licensed by the licensor. Formally, given a sentence s, the licensor l_i from the detection step, and the antecedent candidate a_j; the noun ellipsis resolution task can be defined as follows:

$$f(a_{\mathrm{j}}, l_{\mathrm{i}}, s) \rightarrow \{0, 1\}$$

where 1 denotes that the antecedent candidate a_j is the actual resolution of the ellipsis licensed by l_i, and 0 otherwise.

Embeddings Similar to the detection step, we take pretrained fastText word embeddings for the licensor, antecedent candidate and context, and sum pool to obtain a single vector. In case of BERT, we separate the sentence, the licensor and the antecedent candidate with a [SEP] token and follow the same steps as in the detection step.

Manual Syntactic and Semantic Features We use POS tags of the licensor and modifier of the antecedent as our syntactic features and cosine similarity between their POS tags as our semantic features, following (Khullar et al., 2019). We concatenate these features to the embeddings to explore the efficacy of adding manual features on resolution.

4 Experiments

We choose Naive Bayes and linear SVM as our statistical models, and Multilayer Perceprton (MLP) and bidirectional Long Short Term Memory (bi-LSTM) network as our neural models. For the

Subtask	Classification	Representation	Precision	Recall	F1-Score
Detection	Naive Bayes	FT	0.4192	0.5807	0.4869
		FT+F	0.4923	0.6275	0.5517
		BERT	0.5511	0.6203	0.5837
		BERT+F	0.5990	0.6213	0.6099
	SVM	FT	0.4502	0.5560	0.4975
		FT+F	0.4824	0.6400	0.5501
		BERT	0.5766	0.6503	0.6112
		BERT+F	0.5799	0.6771	0.6247
	MLP	FT	0.6504	0.6974	0.6731
		FT+F	0.7123	**0.8977**	0.7943
		BERT	0.6999	0.7398	0.7193
		BERT+F	**0.9116**	0.8901	**0.9007**
	bi-LSTM	FT	0.6010	0.6531	0.6260
		FT+F	0.6901	0.8745	0.7714
		BERT	0.7001	0.7503	0.7243
		BERT+F	0.8655	0.8163	0.8402

Table 1: Precision, Recall and F1-Score values of different statistical and neural models on the noun ellipsis detection task. FT stands for fastText and +F denotes concatenation of manual features to the embeddings. Values in bold depict best performance.

detection task, we take the annotated 946 positive samples (exophoric) and randomly choose 946 negative samples. Similarly, for the resolution task, we take 438 positive samples (endophoric) and 438 randomly chosen negative samples. We perform a standard 70-10-20 split to obtain the train, development and test set respectively, and follow the 5-fold cross validation procedure to capture both classes properly in each case. For MLP, we take a simple, two-layer feedforward network (FFNN) or two layers of multiple computational units interconnected in a feed-forward way without loops. We have a single hidden layer with 768 neurons and a sigmoid function. A unidirectional weight connection exists between the two successive layers. The classification decision is made by turning the input vector representations of a word with its context into a score. The network has a softmax output layer. For bi-LSTM, we have embedding layer, time-distributed translate layer, Bi-LSTM (RNN) layer, batch normalization layer, dropout layer and prediction layer. The activation used is Softmax. The loss function is calculated with cross entropy. We train in batch sizes of 16 and early stopping with max epochs of 100. In early stopping the patience is kept to be 10 and the optimizer used is Adam. We use default values for the learning rate. We use Keras (Chollet et al., 2015) for coding the models.

5 Results

We evaluate the performance of our models in terms of F1-score, computed by taking an average F1-scores obtained from the 5-folds results. We experiment with sixteen models each for the noun ellipsis detection and resolution. The results on the testset for Precision, Recall and F1-Score values are presented in Table.2. As expected, the neural models perform significantly better than the statistical ones for both the subtasks. Our experiments show that for the detection task, BERT embeddings with a simple MLP gives best scores. This is expected because, BERT currently provides the most powerful contextual word representations, using 12 separate attention mechanism for each layer, where, at each layer, each token can focus on 12 distinct aspects of other tokens. Since Transformers (Vaswani et al., 2017) use many distinct attention heads (12*12=144 for the base BERT model), each head can focus on a different kind of constituent combinations, making BERT broadly attending over the whole sentence. In our task, the

Subtask	Classification	Representation	Precision	Recall	F1-Score
Resolution	Naive Bayes	FT	0.2992	0.5112	0.3775
		FT+F	0.3379	0.5750	0.4257
		BERT	0.3919	0.5503	0.4578
		BERT+F	0.4558	0.6230	0.5264
	SVM	FT	0.4001	0.4960	0.4429
		FT+F	0.5024	0.6040	0.5485
		BERT	0.4966	0.6003	0.5435
		BERT+F	0.5372	0.6611	0.5927
	MLP	FT	0.5901	0.6577	0.6221
		FT+F	0.7005	0.8122	0.7522
		BERT	0.6951	0.7798	0.7350
		BERT+F	0.8015	0.8531	0.8265
	bi-LSTM	FT	0.6009	0.6321	0.6161
		FT+F	0.7029	0.8044	0.7502
		BERT	0.7195	0.7383	0.7288
		BERT+F	**0.8652**	**0.8166**	**0.8401**

Table 2: Precision, Recall and F1-Score values of different statistical and neural models on the noun ellipsis resolution subtask. FT stands for fastText and +F denotes concatenation of manual features to the embeddings.

BERT with MLP model is robust and efficiently makes generalisations on the syntactic and semantic dependency between the elided noun and the pre-modifiers and modifiers (licensors). For the resolution task, however, bi-LSTMs work better than MLP. Unlike MLPs, Bi-LSTMs take into consideration left to right and right to left context, capturing long range dependencies in a sentence and, hence, are better suited for handling the resolution of a cohesive discourse device like ellipsis where the distance between elided phrase and antecedent can be several words. The sufficient neurons in the hidden layer with sigmoidal function ensures the MLP approximate the nonlinear relationships between the input and output, but they are not innately designed to capture temporal relationships within a sentence. Hence, although they perform well for a task like detection that needs local information, they are outperformed by bi-LSTMs on the resolution task that requires capturing a deeper relationship between the antecedent and the elided noun. We also note that manual feature addition boosts results greatly for all models, highlighting that ellipsis is a syntactically constrained phenomenon. We finally integrate the best models for each subtask into an end-to-end pipeline, as in Figure 2. Now, instead of the gold vectors (from the annotations), the resolution model is fed the ouput licensor vector from the detection model. This obviously results into error propagation into the second model, and lowers the precision value to 82.52%, recall to 78.66% and consequently, the F1-score to 80.55% of the final system. The error in the final system comes from failing to detect actual licensors, wrongly identifying non-licensor words and correct licensor detection but failed antecedent resolution. We run our final system on the curated dataset prepared by (Khullar et al., 2019) and compare the results with their rule-based approach. As expected, this model improves the F1-score by 16.55% for noun ellipsis detection and 14.97% for noun ellipsis res-

Curated Dataset	**P**	**R**	**F**
Detection			
(Khullar et al., 2019)	69.15	85.53	76.47
MLP, BERT+F	91.72	94.32	93.02
Resolution			
(Khullar et al., 2019)	78.79	63.41	70.27
bi-LSTM, BERT+F	87.01	83.54	85.24

Table 3: Precision (**P**), Recall (**R**) and F1-Score (**F**) values (%) of the rule-based approach by (Khullar et al., 2019) and the neural model presented in this paper.

olution. See Table 3. The even higher accuracy on the curated dataset can be explained by the nature of the sentences in this dataset which are from textbooks, and, hence, free of grammatical errors, etc. – resulting into improved parser performance in the pre-processing step. Although, the presented models achieve high scores on both the tasks separately and in the pipeline process, the results can be further improved with hyper-parameter tuning and additional regularization.

6 Conclusion

We explored statistical and neural models for noun ellipsis detection and resolution, presenting a strong results for this task. As expected, neural classifiers perform significantly better than the statistical with the same input representation. As with several other NLP tasks, the contextual nature of BERT is useful for noun ellipsis resolution too, making robust predictions with simple neural classifiers. Finally, addition of manual features boosts the performance of all classifiers including those that use BERT, highlighting that ellipsis is a syntactically constrained phenomenon.

References

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

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of vp ellipsis. *Language Resources and Evaluation*, 45(4):463–494.

Wei Chen. 2016. The motivation of ellipsis. *Theory and Practice in Language Studies*, 6(11):2134–2139.

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

Sandra Chung, William Ladusaw, and James McCloskey. 2010. Sluicing (:) between structure and inference. *In Representing language: Essays in honor of Judith Aissen*.

Jeroen van Craenenbroeck and Jason Merchant. 2013. Ellipsis phenomena. In *The Cambridge Handbook of Generative Syntax*, pages 701–745. Cambridge University Press.

Mary Dalrymple, Stuart M. Shieber, and Fernando C. N. Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Kian Kenyon Dean, Jackie Chi Kit Cheung, and Doina Precup. 2016. Verb phrase ellipsis resolution using discriminative and margin-infused algorithms. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, page 1734–1743.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Lyn Frazier. 2008. Processing ellipsis: A processing solution to the undergeneration problem? In *Proceedings of the 26th West Coast Conference on Formal Linguistics*.

Tilbe Goksun, Tom W. Roeper, Kathy Hirsh-Pasek, and Roberta Michnick Golinkoff. 2010. From noun-phrase ellipsis to verbphrase ellipsis: The acquisition path from context to abstract reconstruction.

Christine Gunther. 2011. Noun ellipsis in english: adjectival modifiers and the role of context. *The structure of the noun phrase in English: synchronic and diachronic explorations*, 15(2):279–301.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 1976. Cohesion in english. page 76.

Daniel Hardt. 1992. An algorithm for vp ellipsis. page 9–14. In Proceedings of the 30th annual meeting on Association for Computational Linguistics.

Daniel Hardt. 1998. Improving ellipsis resolution with transformation-based learning. AAAI fall symposium.

Daniel Hardt. 1999. Dynamic interpretation of verb phrase ellipsis. *Linguistics and Philosophy*, 22(2):187–221.

Jerry R. Hobbs and Andrew Kehler. 1997. A theory of parallelism and the case of vp ellipsis. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, pages 394–401, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Nina Hyams, Victoria Mateu, and Lauren Winans. 2017. Ellipsis meets wh-movement: sluicing in early grammar.

Kyle Johnson. 2001. What vp ellipsis can do, and what it can't, but not why. pages 439–479.

Payal Khullar, Allen Anthony, and Manish Shrivastava. 2019. Using syntax to resolve npe in english. In *Proceedings of Recent Advances in Natural Language Processing*, pages 535–541.

Payal Khullar, Kushal Majmundar, and Manish Shrivastava. 2020. Noel: An annotated corpus for noun ellipsis in english. In *Language Resources Evaluation Conference*.

Nayoun Kim, Laurel Brehm, and Masaya Yoshida. 2019. The online processing of noun phrase ellipsis and mechanisms of antecedent retrieval. *Language, Cognition and Neuroscience*, 34(2):190–213.

Shalom Lappin. 1996. The interpretatin of ellipsis. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 145–176. Blackwell.

Ekaterina Lapshinova-Koltunski, Christian Hardmeier, and Pauline Krielke. 2018. Parcorfull: a parallel corpus annotated with full coreference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Charlotte Lindenbergh, Angeliek van Hout, and Bart Hollebrandse. 2015. Extending ellipsis research: The acquisition of sluicing in dutch. *BUCLD 39 Online Proceedings Supplement*, 39.

Chin-Ting Liu, Li-mei Chen, Yu-Ching Lin, Chia-Fang Cheng, and Hui-chen Chang. 2016a. Speech intelligibility and the production of fricative and affricate among Mandarin-speaking children with cerebral palsy. In *Proceedings of the 28th Conference on Computational Linguistics and Speech Processing (ROCLING 2016)*, pages 153–163, Tainan, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Zhengzhong Liu, Edgar Gonzalez, and Dan Gillick. 2016b. Verb phrase ellipsis detection using automatically parsed text. pages 32–40.

Anne Lobeck. 1995. *Functional Heads, Licensing, and Identification*. Oxford University Press.

Marjorie McShane and Petr Babkin. 2016. Detection and resolution of verb phrase ellipsis. *Linguistic Issues in Language Technology*, 13(1).

Marjorie McShane, Sergei Nirenburg, and Petr Babkin. 2015. Sentence trimming in service of verb phrase ellipsis resolution. In *EAPCogSci*.

Katrin Menzel. 2017. *Understanding English-German contrasts: a corpus-based comparative analysis of ellipses as cohesive devices*. Ph.D. thesis, Universitat des Saar- landes, Saarbrucken.

Katrin Menzel and Ekaterina Lapshinova-Koltunski. 2014. Kontrastive analyse deutscher und englischer kohäsionsmittel in verschiedenen diskurstypen. *tekst i dyskurs - Text und Diskurs. Zeitschrift der Abteilung für germanistische Sprachwissenschaft des Germanistischen Instituts Warschau*.

Jason Merchant. 2004. Fragments and ellipsis. *Linguistics and Philosophy*, 27(6):661–738.

Jason Merchant. 2010. *Three Kinds of Ellipsis: Syntactic, Semantic, Pragmatic?*

Ruslan Mitkov. 1999. *Anaphora Resolution*. Oxford University Press.

Leif Arda Nielsen. 2003. Using machine learning techniques for vpe detection.

Leif Arda Nielsen. 2004a. Robust VPE detection using automatically parsed text. In *Proceedings of the ACL Student Research Workshop*, pages 49–54, Barcelona, Spain. Association for Computational Linguistics.

Leif Arda Nielsen. 2004b. Verb phrase ellipsis detection using automatically parsed text.

Dongwoo Park. 2017. *When does ellipsis occur, and what is elided?* PhD dissertation, University of Maryland.

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.

John Robert Ross. 1967. *Constraints on variables in syntax*. Massachusetts Institute of Technology.

Alain Rouveret. 2012. Vp ellipsis, phases and the syntax of morphology. *Natural Language & Linguistic Theory*, 30(3):897–963.

Mamoru Saito, Jonah Lin, and Keiko Murasugi. 2008. Nominal-ellipsis and the structure of noun phrases in chinese and japanese. *Journal of East Asian Linguistics*, 17.

Sebastian Schuster, Joakim Nivre, and Christopher D. Manning. 2018. Sentences with gapping: Parsing and reconstructing elided predicates. *ArXiv e-prints*.

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 *NeurIPS*.

Frank Wijnen, Tom W. Roeper, and Hiske van der Meulen. 2003. Discourse binding: Does it begin with nominal ellipsis?

Wei-Nan Zhang, Yue Zhang, Yuanxing Liu, Donglin Di, and Ting Liu1. 2019. A neural network approach to verb phrase ellipsis resolution.

MRC Examples Answerable by BERT without a Question are Less Effective in MRC Model Training

Hongyu Li[†], **Tengyang Chen**[†], **Shuting Bai**[†], **Takehito Utsuro**[†], **Yasuhide Kawada**[‡]
[†]Graduate School of Systems and Information Engineering, University of Tsukuba, Japan
[‡]Logworks Co., Ltd., Japan

Abstract

Models developed for Machine Reading Comprehension (MRC) are asked to predict an answer from a question and its related context. However, there exist cases that can be correctly answered by an MRC model using BERT, where only the context is provided without including the question. In this paper, these types of examples are referred to as "easy to answer", while others are as "hard to answer", i.e., unanswerable by an MRC model using BERT without being provided the question. Based on classifying examples as answerable or unanswerable by BERT without the given question, we propose a method based on BERT that splits the training examples from the MRC dataset SQuAD1.1 into those that are "easy to answer" or "hard to answer". Experimental evaluation from a comparison of two models, one trained only with "easy to answer" examples and the other with "hard to answer" examples demonstrates that the latter outperforms the former.

1 Introduction

The Machine Reading Comprehension (MRC) task locates the best corresponding natural language answer when provided a question and its related context. In recent years, MRC models using neural networks have been proposed for SQuAD (Pranav et al., 2016, 2018), which is a large-scale, high-quality English MRC dataset. Most recent neural network based MRC models have outperformed human performance (Devlin et al., 2019).

Among those existing work, to analyze the difficulty of several popular MRC benchmarks such as bAbI (Weston et al., 2016), SQuAD (Pranav et al., 2016), CBT (Hill et al., 2016), CNN (Hermann et al., 2015) and Who-did-What (Onishi et al., 2016), Kaushik and Lipton (2018) established sensible baselines for these datasets, and found that question-only and context-only (which is called *passage-only* in Kaushik and Lipton (2018)) models often performs surprisingly well. In particular, context-only models achieve over 50% accuracy on 14 out of 20 bAbI tasks, and as for CBT, only the last one of the 20 sentences provided as a context is necessary to achieve a comparable accuracy. They also indicated that SQuAD is designed more carefully than other datasets and achieved F1 scores of only 4% and 14.8% respectively on question-only and context-only models, which are relatively lower. Kaushik and Lipton (2018) demonstrated that published MRC datasets should characterize the level of difficulty, and specifically, the extent to which questions and contexts are essential. Moreover, Kaushik and Lipton (2018) also claimed that follow-up papers reporting improvements ought to report performance both on the full task and variations omitting questions and contexts. In view of the point demonstrated in Kaushik and Lipton (2018), we concentrate more on the difficulty of every single MRC example, and aim to split the examples into easy ones and hard ones.

Given the MRC dataset SQuAD1.1 (where each MRC example denoted as the tuple $\langle Q, C, A\rangle$ of the question Q, the context C, and the answer A) and the fine-tuned MRC model using BERT (Devlin et al., 2019), there exist *context-only* examples that can be correctly answered, where only the context is provided without including the question. By focusing on this fact, this paper proposes a method that splits the MRC examples into binary classes of "easy to answer" or "hard to answer". A 10-fold cross-validation was applied on approximately 87,600 SQuAD1.1 training examples comprised of 12,500 "easy to answer" and 75,000 "hard to answer" classes. From the comparison of the two classes, the followings are two significant findings. (1) Based

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 139–145

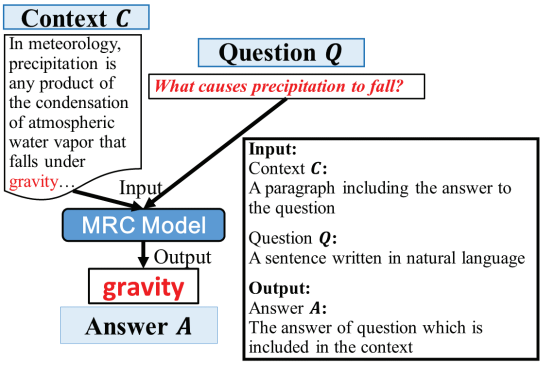


Figure 1: An MRC Model using Neural Networks

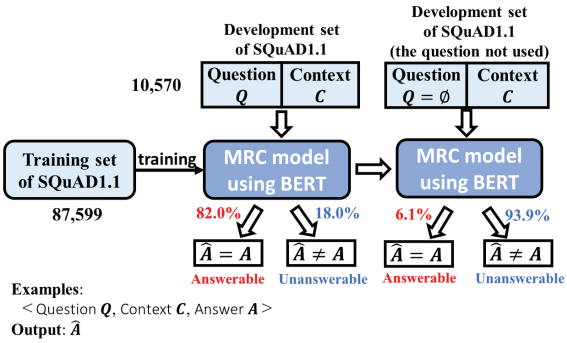


Figure 2: Detecting MRC Examples as Answerable without an Input Question

on the performance of training the BERT MRC models with 12,500 "easy to answer" and "hard to answer" examples each, the model trained with "hard to answer" examples outperformed the other. (2) An analysis of the position distribution of answers A within the context C, answers from the "easy to answer" MRC example class tend to be located around the beginning of the context compared with those from the "hard to answer" MRC example class.

2 Machine Reading Comprehension using Neural Networks

Figure 1 shows the framework of MRC models that use neural networks. In the MRC model, when a question and context are input, the starting and ending positions of the answer with respect to the question within the context are predicted.

Let ts be a set of test examples with each example denoted as $s(\in ts)$. Here, s is represented as $s = \langle Q, C, A\rangle$. Also, if a set of examples for training MRC models is denoted as tr, then the corresponding model is represented as $m(tr)$. Then, the answer $\widehat{A}$ predicted from an input test example s with the trained MRC model $m(tr)$ is denoted as

$$\widehat{A} \;=\; answer\Big(m(tr),\; s\Big)$$

A Boolean predicate $answerable$ classifies if the given test example s is "answerable" or "unanswerable" by the trained MRC model $m(tr)$, and is defined according to if the predicted answer $\widehat{A}$ is the same as the reference answer A as

$$answerable\Big(m(tr),\; s\Big) \;=\; \begin{cases} 1 & (\widehat{A} = A) \\ 0 & (\widehat{A} \neq A) \end{cases}$$

3 Machine Reading Comprehension using BERT

As described above, the MRC models are trained and tested with 87,600 training examples from the MRC dataset SQuAD1.1. The fine-tuning module for machine reading comprehension[1] was applied to the pre-trained Multilingual Cased model[2] of BERT.[3]

4 MRC Examples Answerable without a Question

First, we evaluate the BERT MRC model trained with the 87,600 training examples against 10,570 development examples from of SQuAD1.1. As shown in Figure 2, we compare the two cases of *with* or *without* the question for each of these development examples. Let s denote one of the 10,570 development examples represented as $s = \langle Q, C, A\rangle$, as before, which are each evaluated with the BERT MRC model trained with the 87,600 examples. From these results, 82% are correctly answered and classified as "answerable" with the remaining 18% incorrectly answered and classified as "unanswerable." Next, let s' denote an MRC example obtained by replacing the question Q from each example s of the development examples with an empty question $Q' = \emptyset$ represented as $s' = \langle Q' = \emptyset, C, A\rangle$. After evaluating the trained BERT MRC model with each example s', as shown on the right of Figure 2, 6.1% are

[1] `run_squad.py`, with the number of epochs as 2, batch size as 8, learning rate as 0.00003, and the maximum sequence length as 320.

[2] Trained with 104 languages, available from https://github.com/google-research/bert/blob/master/multilingual.md.

[3] The TensorFlow version of BERT (https://github.com/google-research/bert) is used.

		Training set	
		Questions used tr_i	Questions not used $tr_i{}^{Q=\emptyset}$
Test set	questions used $s\ (\in ts_i)$	Answerable examples $a^1(ts_i)$ Unanswerable examples $ua^1(ts_i)$ $\lvert a^1\rvert = 60,668$ $\lvert ua^1\rvert = 26,931$	Answerable examples $a^2(ts_i)$ Unanswerable examples $ua^2(ts_i)$ $\lvert a^2\rvert = 11,498$ $\lvert ua^2\rvert = 76,101$
	questions not used $s^{Q=\emptyset}$ $(s \in ts_i)$	Answerable examples $a^3(ts_i)$ Unanswerable examples $ua^3(ts_i)$ $\lvert a^3\rvert = 3,682$ $\lvert ua^3\rvert = 83,917$	Answerable examples $a^4(ts_i)$ Unanswerable examples $ua^4(ts_i)$ $\lvert a^4\rvert = 10,008$ $\lvert ua^4\rvert = 77,591$

Table 1: Splitting the MRC Examples into "Answerable" and "Unanswerable" Examples with the Corresponding Statistics

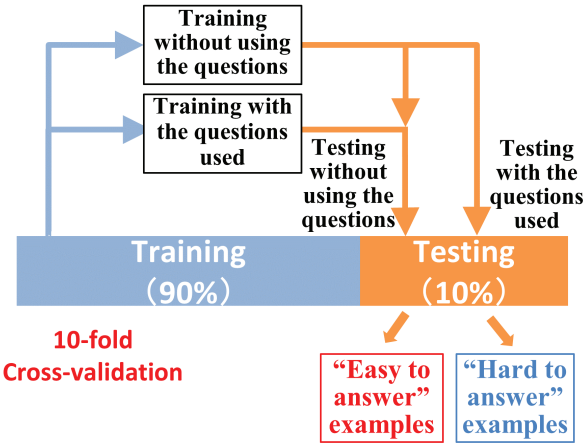


Figure 3: Splitting the MRC Examples into "Easy to Answer" and "Hard to Answer" Classes with 10-fold Cross-Validation

correctly answered and classified as "answerable" even without being provided an appropriate question, while the remaining 93.9% are incorrectly answered and classified as "unanswerable."

5 Splitting MRC Examples into "Easy to Answer" and "Hard to Answer" Classes

Following the procedure for detecting MRC examples as answerable without a question demonstraed in the previous section, we similarly split the 87,600 SQuAD1.1 training examples into "easy to answer" and "hard to answer" classes. As illustrated in Figure 3, the process designates 10% of the examples as "easy to answer" and "hard to answer" classes for testing through one fold of 10-fold cross-validation, which is repeated ten times, resulting in 12,500 "easy to answer" and 75,100 "hard to answer" classes. From this, we obtain the following three types of evaluation results.

(i) The MRC model is trained with the training examples that include questions used as they are, while the trained MRC model is evaluated against the MRC test examples without questions.

(ii) The MRC model is trained with training examples that do not include questions, with

(ii-a) the trained MRC model is evaluated against the MRC test examples without questions, or

(ii-b) the trained MRC model is evaluated against the MRC test examples with questions.

Detailed Procedure

The SQuAD1.1 dataset is composed of approximately 100,000 MRC examples that use 23,215 paragraphs extracted from 536 Wikipedia articles as context. With these contexts, questions and answers are annotated through crowdsourcing to generate the complete 100,000 MRC example set. From these examples, we apply N-fold cross-validation (N=10 in this paper) to the set U of the MRC training examples collected from 442 out of the 536 Wikipedia articles.

Before the N-fold cross-validation, we first divide the 442 Wikipedia articles into disjoint N subsets. From the i-th ($i = 1, \ldots, N$) subset of Wikipedia articles, we obtain the i-th test set ts_i of the MRC examples, and the i-th training set of the MRC examples is obtained as the set tr_i of the remaining MRC examples. Then, the set U of the complete SQuAD1.1 training example set is represented as

$$U = \bigcup_{i=1,\ldots,N} ts_i \quad \left(ts_i \cap ts_j = \emptyset \ \ (i \neq j) \right)$$

As shown in Table 1, from the i-th training set tr_i of the MRC examples, each of which contains a question, another training set $tr_i{}^{Q=\emptyset}$ of the MRC examples is obtained by removing the question Q from each example. So, each MRC example in the obtained training set $tr_i{}^{Q=\emptyset}$ now has an empty question. Similarly, from a test MRC example s in the i-th test set ts_i of the MRC examples that contains a question, another test MRC example $s^{Q=\emptyset}$ is obtained by removing its question Q from s. So, the obtained test MRC example $s^{Q=\emptyset}$ has an empty question. By pairing the two training sets tr_i and $tr_i{}^{Q=\emptyset}$ from the MRC examples and the two test MRC examples s and $s^{Q=\emptyset}$, as shown

in Table 1, a resulting four pairs of training sets from the MRC examples and a test MRC example can be examined as to if the MRC model trained with the designated training set is "answerable" or "unanswerable" given the designated test MRC example. Finally, in each of these four pairs, the set ts_i of the test MRC examples is split into the set $a^{\alpha}(ts_i)$ of answerable test MRC examples and $ua^{\alpha}(ts_i)$ of unanswerable test MRC examples, according to $(\alpha = 1, 2, 3, 4)$

$$a^1(ts_i) = \left\{ s \in ts_i \middle| answerable\Big(m(tr_i), s\Big) = 1 \right\}$$

$$a^2(ts_i) = \left\{ s \in ts_i \middle| answerable\Big(m({tr_i}^{Q=\emptyset}), s\Big) = 1 \right\}$$

$$a^3(ts_i) = \left\{ s \in ts_i \middle| answerable\Big(m(tr_i), s^{Q=\emptyset}\Big) = 1 \right\}$$

$$a^4(ts_i) = \left\{ s \in ts_i \middle| answerable\Big(m({tr_i}^{Q=\emptyset}), s^{Q=\emptyset}\Big) = 1 \right\}$$

$$ua^{\alpha}(ts_i) = ts_i - a^{\alpha}(ts_i) \ (\alpha = 1, 2, 3, 4)$$

The sets $a^{\alpha}(ts_i)$ $(\alpha = 1, 2, 3, 4)$ of "answerable" test MRC examples are obtained by evaluating the MRC model trained with the training sets tr_i (with questions) or ${tr_i}^{Q=\emptyset}$ (without questions) against s (with a question) or $s^{Q=\emptyset}$ (without a question). We define the set E of "easy to answer" MRC examples as the union of the three sets $a^{\alpha}(ts_i)$ $(\alpha = 2, 3, 4)$ of "answerable" test MRC examples. For these, we collect the "answerable" test MRC examples over the cases with questions removed either from the training or test MRC examples ($a^1(ts_i)$ is excluded because the questions are used in both the training and test MRC examples). The set H of "hard to answer" MRC examples is subsequently defined as the complement set of E.[4]

Consequently, as shown in Table 2, the set U of the complete SQuAD1.1 training examples is split into the set E of 12,487 "easy to answer" examples and the set H of 75,112 "hard to answer" examples. Figure 4 compares the distributions of positions of answers within the contexts as a ratio of the "start position of an answer" to the "length of context". These results indicate that the answers of the "easy to answer" MRC examples tend to be located near the beginning of the context as compared with those of the "hard to answer" MRC examples.[5] We repeat this splitting procedure ten times and compare the numbers of "easy to answer" and "hard to answer" examples, where we have almost the same results as we report in this section. For examples of the "easy to answer" MRC examples, Figure 5 provides two cases, one of which is a typical "easy to answer" with its answer located exactly at the beginning of the context, and a second as the opposite class with its answer located exactly at the end of the context.

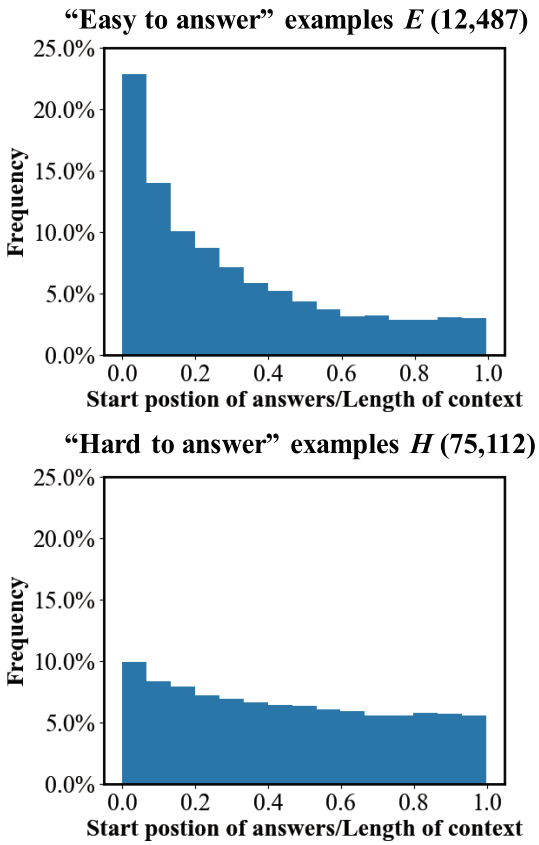


Figure 4: Distributions of the Positions of Answers in the Contexts as the Ratio of "start position of an answer" / "length of context"

6 Effectiveness of "Hard to Answer" Examples in MRC Model Training

We next evaluate the effectiveness of "hard to answer" and "easy to answer" MRC examples based

[4] Over the set U of the complete SQuAD1.1 training examples, the set a^{α} of "answerable" examples and the set ua^{α} of "unanswerable" examples are defined as $a^{\alpha} = \bigcup_{i=1,\ldots,N} a^{\alpha}(ts_i)$, $ua^{\alpha} = U - a^{\alpha}$ $(\alpha = 1, 2, 3, 4)$, where the number of examples in each set is provided in Table 1.

[5] We also compare the context length between the "easy to answer" and "hard to answer" examples, where we did not detect any significant difference.

Question Q : What began as an almost exclusively linguistic and philological enterprise?
Context C : ***Slavic studies*** began as an almost exclusively linguistic and philological enterprise. As early as 1833, Slavic languages were recognized as Indo-European.
Answer A : ***Slavic studies***

Question Q : What is a significant early architectural canonical type?
Context C : Texts on architecture have been written since ancient time. These texts provided both general advice and specific formal prescriptions or canons. Some examples of canons are found in the writings of the 1st-century BCE Roman Architect Vitruvius. Some of the most important early examples of canonic architecture are ***religious***.
Answer A : ***religious***

Figure 5: Two Sample "Easy to Answer" MRC Examples

Training set	Number of examples
U: training set of SQuAD1.1	87,599
H: "hard to answer" examples M: examples randomly sampled from U	75,112
E: "easy to answer" examples H_{sml}: examples randomly sampled from H M_{sml}: examples randomly sampled from U	12,487

Table 2: Number of Examples in Each Training Set

on the performance of each class when used for the MRC model training. The sets shown in Table 2 are evaluated as the MRC model training examples. In addition to the sets H and E, we evaluate a set M comprised of $|H| = 75,112$ MRC examples randomly sampled from U and sets H_{sml} and M_{sml} of $|E| = 12,487$ MRC examples randomly sampled from H and U, respectively. The sets H_{sml} and M_{sml} are intended to directly compare the effectiveness of the "easy to answer", "hard to answer", and (randomly sampled) SQuAD1.1 training examples by restricting the numbers of the training examples to be the same. The set M is also intended to directly compare the effectiveness of the "hard to answer" and SQuAD1.1 training examples by restricting the numbers of training examples to be the same. All these sets are used to fine-tune the BERT pre-trained model on the MRC task, and the development set of SQuAD1.1

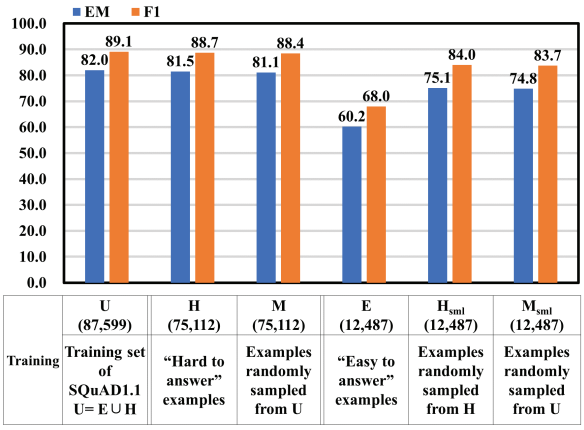


Figure 6: Evaluation Results on the Development Set of SQuAD1.1 where EM is an Exact Match, and F1 is the Macro-Average of the F1 Score per Example

is used as the test set for each evaluation. For the evaluation measures, we utilize the exact match (EM), which is defined as the rate of examples with a predicted answer that exactly matches the reference answer. The macro average of the F1 score is calculated from the precision and recall between the token sequences of the predicted and reference answers.

Figure 6 compares the performance of the five MRC training examples, and Figure 7 presents the learning curves with the training examples of the sets E and H_{sml} of Table 2 used against the development set of SQuAD1.1 as the test set. From both results, the set H_{sml} outperforms the set E with a statistically significant (p<0.01) difference, suggesting that the "hard to answer" examples are effective in MRC model training[6]. Unfortunately, however, Figure 6 also presents that the performance of H_{sml} is almost comparable with that of M_{sml}. From this result, our definite future work includes inventing a technique of automatic selection of MRC training examples from the set U of the complete SQuAD1.1 training example set, which outperform those of the same size randomly sampled from U.

Also, although we omit the detailed evaluation results, in addition to BERT, we also applied SpanBERT (Joshi et al., 2020)[7] (base & cased) and XLNet (Yang et al., 2019)[8] (XLNet-Large, Cased) and obtained the similar results regarding both of

[6]We repeat the splitting procedure and the evaluation procedure ten times, where we have almost the same evaluation results we report in this section.

[7]https://github.com/facebookresearch/SpanBERT

[8]https://github.com/zihangdai/xlnet

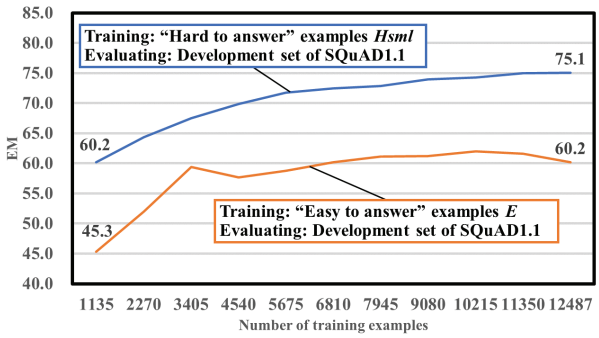


Figure 7: Comparison of the Learning Curves for the Exact Match (EM)

(1) the model trained with "hard to answer" examples outperformed that trained with "easy to answer" ones, and (2) answers from the "easy to answer" MRC example class tend to be located around the beginning of the context compared with those from the "hard to answer" MRC example class.

7 Related Work

Swayamdipta et al. (2020) proposed a general framework of identifying three regions, namely, ambiguous, easy to learn, and hard to learn within a dataset, and applied the framework to several tasks such as natural language inference and sentence-level machine reading comprehension. It is concluded that ambiguous instances are useful for high performance, easy to learn instances are aid optimization, and hard to learn instances correspond to data errors. Following the conclusions of Swayamdipta et al. (2020), our future work include applying the framework of Swayamdipta et al. (2020) to the tasks of machine reading comprehension studied in this paper and investigating the difference of our notion of "easy to answer" / "hard to answer" and their notion of "easy to learn" / "hard to learn." Among other related work, Sugawara et al. (2018) studied splitting 12 MRC datasets into easy and hard subsets according to two types of simple lexical based heuristics and showed that the performance against easy subsets were lower than the whole datasets. Min et al. (2018) also studied to select minimal set of sentences within the context of existing MRC datasets to answer the MRC question.

In the task of recognizing textual entailment that classifies the relation between a pair of two sentence as a premise and hypothesis, Tsuchiya (2018) compared two of the "Recognizing Textual Entailment" datasets, SICK (Bowman et al., 2015) and SNLI (Marelli et al., 2014). Tsuchiya reported that the cases of SNLI had the correct textual entailment labels predicted when only the hypothesis sentence was provided and without the premise sentence. However, Tsuchiya (2018) also pointed out that, a hidden bias in the SNLI corpus caused much of the high accuracy achieved by the neural network based models that were trained with SNLI.

Developing machine reading comprehension datasets requires an expensive and time-consuming effort to manually create questions from paragraphs and extract spans of text from each paragraph to represent the answer to each question. The approach of active learning, in which the key idea is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns (Settles, 1995, 2010), could be applied to reduce the cost of developing MRC datasets. While there exists no previous study that applies the active learning technique for machine reading comprehension task, other work applied the technique to reduce the cost of developing datasets for other NLP tasks (Sener and Savarese, 2018; Chen et al., 2019), image classification (Beluch et al., 2018; Fang et al., 2017), as well as other machine learning tasks, such as predicting molecular energetics in the field of chemistry (Smith et al., 2018).

8 Conclusion

We proposed a method based on BERT (Devlin et al., 2019) that splits the training examples from the MRC dataset SQuAD1.1 into classes of "easy to answer" and "hard to answer." Experimental evaluations of comparing the two models, one of which is trained only with the "easy to answer" examples and the other with the "hard to answer" examples, demonstrate that the latter outperformed the former. Future work includes applying the analysis procedure of this paper to several popular MRC benchmark datasets other than SQuAD (Pranav et al., 2016) and investigating whether the similar results are obtained. We also work on deeper analysis of the characteristics of "easy to answer" / "hard to answer" examples to find out features that are related to the disparity of training effectiveness.

References

W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. 2018. The power of ensembles for active learning in image classification. In *Proc. 31st CVPR*, pages 9368–9377.

S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. 20th EMNLP*, pages 632–642.

X. C. Chen, A. Sagar, J. T. Kao, T. Y. Li, C. Klein, S. Pulman, A. Garg, and J. D. Williams. 2019. Active learning for domain classification in a commercial spoken personal assistant. In *Proc. 20th INTERSPEECH*, pages 1478–1482.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, pages 4171–4186.

M. Fang, Y. Li, and T. Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proc. EMNLP*, pages 595–605.

K. M. Hermann, T. Kociskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. 28th NIPS*, pages 1693–1701.

F. Hill, A. Bordes, S. Chopra, and J. Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proc. 4th ICLR*, pages 1–13.

M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

D. Kaushik and Z. C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proc. EMNLP*, pages 5010–5015.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proc. 9th LREC*, pages 216–223.

S. Min, V. Zhong, R. Socher, and C. Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proc. 56th ACL*, pages 1725–1735.

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proc. EMNLP*, pages 2230–2235.

R. Pranav, Z. Jian, L. Konstantin, and L. Percy. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. EMNLP*, pages 2383–2392.

R. Pranav, J. Robin, and L. Percy. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proc. 56th ACL*, pages 784–789.

O. Sener and S. Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *Proc. 6th ICLR*, pages 1–13.

B. Settles. 1995. Active learning literature survey. *Science*, 10(3):237–304.

B. Settles. 2010. Active learning literature survey. University of Wisconsin–Madison, Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev, and A. E. Roitberg. 2018. Less is more: Sampling chemical space with active learning. *Journal of Chemical Physics*, 148(241733).

S. Sugawara, K. Inui, S. Sekine, and A. Aizawa. 2018. What makes reading comprehension questions easier? In *Proc. EMNLP*, pages 4208–4219.

S. Swayamdipta, R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith, and Y. Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proc. EMNLP*.

M. Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proc. 11th LREC*, pages 1506–1511.

J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proc. 4th ICLR*, pages 1–14.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Proc. 33rd NeurIPS*, pages 5753–5763.

Text Simplification with Reinforcement Learning using Supervised Rewards on Grammaticality, Meaning Preservation, and Simplicity

Akifumi Nakamachi[†], **Tomoyuki Kajiwara**[‡], **Yuki Arase**[†]
[†]Graduate School of Information Science and Technology, Osaka University
[‡]Institute for Datability Science, Osaka University
[†]{nakamachi.akifumi, arase}@ist.osaka-u.ac.jp
[‡]kajiwara@ids.osaka-u.ac.jp

Abstract

We optimize rewards of reinforcement learning in text simplification using metrics that are highly correlated with human-perspectives. To address problems of exposure bias and loss-evaluation mismatch, text-to-text generation tasks employ reinforcement learning that rewards task-specific metrics. Previous studies in text simplification employ the weighted sum of sub-rewards from three perspectives: grammaticality, meaning preservation, and simplicity. However, the previous rewards do not align with human-perspectives for these perspectives. In this study, we propose to use BERT regressors fine-tuned for grammaticality, meaning preservation, and simplicity as reward estimators to achieve text simplification conforming to human-perspectives. Experimental results show that reinforcement learning with our rewards balances meaning preservation and simplicity. Additionally, human evaluation confirmed that simplified texts by our method are preferred by humans compared to previous studies.

1 Introduction

Text simplification is one of the text-to-text generation tasks that rewrites complex sentences into simpler ones. Text simplification is useful for preprocessing of NLP tasks such as semantic role labeling (Vickrey and Koller, 2008; Woodsend and Lapata, 2014) and machine translation (Štajner and Popović, 2016, 2018). It also has valuable applications such as assisting language learning (Inui et al., 2003; Petersen and Ostendorf, 2007) and helping language-impaired readers (Carroll et al., 1999).

There are two problems in text-to-text generation with an encoder-decoder model: exposure bias and loss-evaluation mismatch (Ranzato et al., 2016; Wiseman and Rush, 2016). The former is that the model is not exposed to its own errors during training. The latter is that while the generated sentence is evaluated as a whole sentence during inference, it is evaluated at the token-level during training. To address these problems, reinforcement learning has been employed in text-to-text generation tasks, such as machine translation (Ranzato et al., 2016) and abstractive summarization (Paulus et al., 2018). These studies use metrics suitable for each task, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), as rewards. Although reinforcement learning based text simplification models (Zhang and Lapata, 2017; Zhao et al., 2020) have used rewards metrics such as SARI (Xu et al., 2016) and FKGL (Kincaid et al., 1975), these metrics do not align with human-perspectives, *i.e.*, human evaluation results (Xu et al., 2016; Sulem et al., 2018; Alva-Manchego et al., 2020).

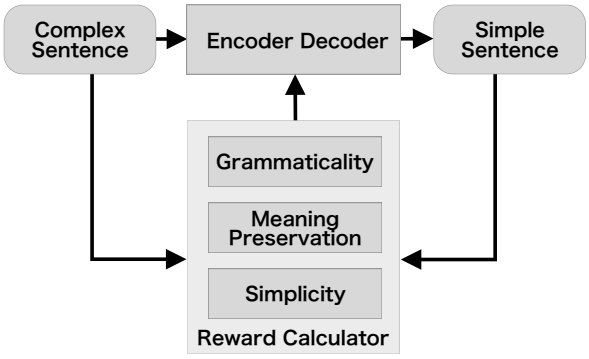


Figure 1: Overview of the reinforcement learning for text simplification.

In this study, we train a text simplification model based on reinforcement learning with rewards that highly agree with human-perspectives. Specifically, we apply a BERT regressor (Devlin et al., 2019) on grammaticality, meaning preservation, and simplicity, respectively, as shown in Figure 1. Experiments on the Newsela dataset (Xu et al., 2015) have shown that reinforcement learning with our rewards balances meaning preservation and simplicity. Further, manual evaluation has shown that

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 146–152

our outputs were preferred by humans compared to previous models.

2 Background: Reinforcement Learning for Text Simplification

Reinforcement learning in text-to-text generation tasks is performed as additional training for pre-trained text-to-text generation models. It is a common technique to linearly interpolate a reward of reinforcement learning and the cross-entropy loss to avoid misleading training because of a large action space (Ranzato et al., 2016; Zhang and Lapata, 2017). We first explain an attention based encoder-decoder model (EncDecA) (Luong et al., 2015) in Section 2.1 and then reinforcement learning for text simplification in Section 2.2.

2.1 Encoder-Decoder Model with Attention

Let $X = (x_1, \cdots, x_{|X|})$ be a source sentence and $Y = (y_1, \cdots, y_{|Y|})$ be its reference sentence. In text simplification, the source and reference are complex and simple sentences, respectively. An encoder takes a source sentence as input, and outputs hidden states. Decoder generates a word distribution at time step $t+1$ from all the encoder hidden states and the series of decoder hidden state $(h_1, \cdots, h_t)$. We generate a sentence $\hat{Y}$ by sampling words from the distribution at each time step.

The objective function for training is averaged cross entropy loss of sentence pairs:

$$\mathcal{L}_{\mathrm{C}} = -\sum_{t=1}^{|Y|} \log P(y_{t+1}|y_{1\cdots t}, X). \quad (1)$$

As the Equation (1) suggests, $y_{1\cdots t}$ is given at training but not at an inference (exposure bias situation). In addition, cross entropy loss cannot be evaluated at a sentence-level (loss-evaluation mismatch).

2.2 Reinforcement Learning

Similar to other text-to-text generation tasks (Ranzato et al., 2016; Paulus et al., 2018), reinforcement learning is applied for text simplification (Zhang and Lapata, 2017; Zhao et al., 2020) to address the problems of exposure bias and loss-evaluation mismatch. In the reinforcement learning step, the pre-trained text-to-text generation model is trained to increase the reward $R(\cdot)$. By employing a reward function that takes the entire sentence $\hat{Y}$ into account, the exposure bias and loss-evaluation mismatch problems are mitigated.

As automatic evaluation metrics for text simplification, BLEU, SARI, and FKGL have been used; however, there has not been a consensus of standard metrics because of their low correlation with human perspectives (Xu et al., 2016; Sulem et al., 2018; Alva-Manchego et al., 2020). Therefore, the previous studies designed rewards from the following three perspectives, based on the standards in manual evaluation for text simplification.

- Grammaticality: This reward assesses the grammatical acceptability of the generated sentence $\hat{Y}$. Previous studies used an neural language model implemented using Long short-term memory (Mikolov et al., 2010; Hochreiter and Schmidhuber, 1997).
- Meaning Preservation: This reward assesses the semantic similarity between the source sentence X and the generated sentence $\hat{Y}$. Zhang and Lapata (2017) used cosine similarity of the sentence representations from a sequence auto-encoder (Dai and Le, 2015). Zhao et al. (2020) used cosine similarity of sentence representations which consists of weighted average of word embeddings (Arora et al., 2017).
- Simplicity: This reward assesses the simplicity of the generated sentence $\hat{Y}$. Zhang and Lapata (2017) used SARI$(X, Y, \hat{Y})$ score, while Zhao et al. (2020) used FKGL$(\hat{Y})$ score.

Among different ways to conduct reinforcement learning, one of the standard approaches used in text simplification is directly maximizing the rewards by the REINFORCE algorithm (Williams, 1992; Ranzato et al., 2016). This approach optimizes the log probability weighted by the expected future reward as the objective function:

$$\mathcal{L}_{\mathrm{R}} = -\sum_{t=1}^{|Y|} r(h_t) \log P(y_{t+1}|y_{1\cdots t}, X), \quad (2)$$

where the expected future reward $r(h_t)$ is estimated using a reward estimator $R(\cdot)$ and a baseline estimator $b(h_t)$ calculated from the hidden state at time step t.

$$r(h_t) = R(\cdot) - b(h_t). \quad (3)$$

Following (Ranzato et al., 2016), the baseline estimator is optimised by minimizing $\|b_t - R(\cdot)\|^2$.

Hashimoto and Tsuruoka (2019) discussed problems in text-to-text generation by reinforcement

learning; the expected future reward estimation is unstable due to the huge action space, which hinders convergence. This is because the action space of text-to-text generation corresponds to the entire target vocabulary, where many words are rarely used for prediction. Therefore, previous studies (Wu et al., 2018; Paulus et al., 2018; Hashimoto and Tsuruoka, 2019) proposed to stabilize the training in reinforcement learning by first pre-training a model with cross-entropy loss, and then adding weighted REINFORCE loss:

$$\mathcal{L} = \lambda \mathcal{L}_R + (1 - \lambda)\mathcal{L}_C. \quad (4)$$

3 BERT-based Supervised Reward

We propose a reward estimator $R(X, \hat{Y})$ consisting of sub-rewards for grammaticality R_{G}, meaning preservation R_{M}, and simplicity R_{S}. These sub-rewards are combined by weighted sum with hyper parameters of δ and ϵ:

$$R(X, \hat{Y}) = \delta R_{\mathrm{G}}(\hat{Y}) + \epsilon R_{\mathrm{M}}(X, \hat{Y}) + (1 - \delta - \epsilon) R_{\mathrm{S}}(\hat{Y}). \quad (5)$$

To achieve a better correlation between each sub-reward and human perspectives, we employ BERT regressors and fine-tune them using manually annotated datasets.

3.1 Implementation Details

For each sub-reward model, we fine-tuned a pre-trained bert-base-uncased model[1] from Hugging Face Transformers library (Wolf et al., 2019). Dropout of 0.2 was applied to all embedding and hidden layers. All the models were optimized using the Adam optimizer (Kingma and Ba, 2015). We linearly decrease learning rate with a warm up in the first $1,000$ training steps. The batch size was 32 sentences. We created a checkpoint for the model at every 100 steps. The training stopped after 10 epochs without improvement in the Pearson correlation measured on the validation set.

Each sub-reward estimator was fine-tuned using the following datasets. Table 1 shows the statistics for each dataset.

Grammaticality We use the GUG dataset[2] (Heilman et al., 2014) for estimating the grammaticality of a sentence. The GUG dataset consists of sentences written by English as the second language learners. Each sentence has four native English speakers assessing grammatical acceptability on a scale of 1 to 4. We estimate the average of these ratings.

	Train	Validation	Test
GUG	$1,518$	747	754
STS-B	$5,749$	$1,500$	$1,379$
Newsela	$94,208$	$1,129$	$1,077$

Table 1: The numbers of sentences in datasets for each sub-reward estimator

Meaning Preservation We use the STS-B dataset[3] (Cer et al., 2017) for estimating the meaning preservation of sentence pairs. The STS-B dataset consists of sentence pairs from multiple sources such as news headlines and image captions. Each sentence pair is evaluated for semantic similarity by five cloud workers on a scale of 0 to 5. We estimate the average of these ratings.

Simplicity We use the Newsela dataset[4] (Xu et al., 2015) for estimating the simplicity of a sentence. The Newsela dataset is a parallel dataset of complex and simple sentences. Each sentence is assigned a U.S. elementary school reading level on a scale of 2 to 12. We follow the data split by Zhang and Lapata (2017). We estimate the grade level of a single sentence using the BERT regressor.

3.2 Intrinsic Evaluation of Rewards

We evaluated how well our sub-reward estimators correlate with human perspectives, compared to previous studies (Zhang and Lapata, 2017; Zhao et al., 2020).

Compared Models We reimplemented the sub-reward estimators introduced in Section 2.2. For R_{G} estimator, we used a 2-layer LSTM language model of 256 hidden dimensions and word embeddings of 300 dimensions. For R_{M} estimator in Zhang and Lapata (2017), we implemented a sequence auto-encoder with bidirectional LSTMs as an encoder. For R_{M} estimator in Zhao et al. (2020), we used 300-dimensional word2vec embeddings[5]

[1] https://huggingface.co/bert-base-uncased

[2] https://github.com/EducationalTestingService/gug-data

[3] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

[4] https://newsela.com/data/

[5] https://code.google.com/archive/p/word2vec/

(Mikolov et al., 2013).

Datasets For grammaticality and meaning preservation, we used the test sets of GUG and STS-B to evaluate the Pearson correlation with human evaluations. For simplicity, a reward estimator should be sensitive to grade levels of sentences with the same meaning, because text simplification intends to preserve the original meaning of the input sentence. Therefore, we extracted pairs of (Y_1, Y_2) of the same source sentence from the Newsela dataset and evaluated the Pearson correlation between the difference of estimated simplicity and the difference of gold-standard grade levels. Note that R_{S} in Zhang and Lapata (2017), *i.e.*, SARI, requires source and reference sentences. Hence, we regarded the simplest sentence of the same source as the reference. We extracted 323 sentence pairs of simplified versions of the same sentence from the Newsela test set.

Results Table 2 shows the evaluation results of each sub-reward estimator. In all perspectives, existing unsupervised sub-reward estimators have little or no correlation with human annotations. As expected, fine-tuning BERT for each task significantly improved the Pearson correlations.

	G	M	S
Zhang's sub-rewards	−0.135	0.041	0.034
Zhao's sub-rewards	−0.135	0.379	0.175
Our sub-rewards	**0.726**	**0.846**	**0.473**

Table 2: Pearson correlation of each sub-reward estimator. Note that G, M, S correspond to grammaticality, meaning preservation, and simplicity, respectively.

4 End-to-End Evaluation on Text Simplification

In this section, we evaluate our rewards on an end-to-end text simplification task using the Newsela dataset[6] shown in Table 1.

4.1 Baseline Encoder-Decoder Model

We implemented and pre-trained the EncDecA model as a common base to add reinforcement learning with rewards of ours and previous studies (Zhang and Lapata, 2017; Zhao et al., 2020). The EncDecA model has a 2-layer LSTM of 256 hidden dimensions for both the encoder and decoder, and attention mechanism by multi-layer perceptron with a layer size of 256. It has word embedding layers of 300 dimensions tying the source, target, and the output layer's weight matrices. Dropout of 0.2 was applied to all embeddings and hidden layers. We used byte-pair encoding[7] (Sennrich et al., 2016) to limit the vocabulary size to 20,000 in addition to the pre-processing by Zhang and Lapata (2017).

The EncDecA model was pre-trained by cross entropy loss with Adam optimizer ahead of reinforcement learning. The batch size was 32 sentences. We created a checkpoint for the model at every 100 steps. In the pre-training, training was stopped after 10 epochs without improvement of SARI score measured on the validation set. However, as the SARI is not stable at the beginning of the training, we ignored checkpoints whose BLEU scores measured on the validation set were less than 21, as suggested by Vu et al. (2018).

4.2 Hyper-Parameter Settings

In reinforcement learning, the hyperparameter λ in Equation (4) was initialized to 0.1, and linearly increased for each iteration until 0.9 during first 10 epochs for stabilizing training process. Following Zhang and Lapata (2017), we trained reinforcement learning models with stochastic gradient descent optimizer with a learning rate of 0.001 and a momentum term of 0.9. Additionally, we trained the baseline estimator with Adam optimizer with a learning rate of 0.001. In the reinforcement learning, training was stopped after 10 epochs without improvement on rewards measured on the validation set.

We set the equal weights to our sub-rewards, *i.e.*, assigned $1/3$ to δ and ϵ in Equation (5), respectively. Tuning of these weights is our future work.

4.3 Results of Automatic Evaluation

The performance of each method is automatically evaluated using the EASSE toolkit[6] by BLEU and SARI. Furthermore, we perform detailed automatic evaluations of grammaticality, meaning preservation, simplicity, and overall quality defined in Equation (5) using our sub-reward estimators.

Table 3 shows the experimental results. In both our reward and existing rewards, reinforcement learning has improved the EncDecA baseline in

[6] We did not experiment with the Simple English Wikipedia because it does not have a detailed, difficulty-by-difficulty rewrite.

[7] https://github.com/google/sentencepiece

	Metrics		Rewards				Human
	BLEU	SARI	G	M	S	O	Avg. Rank
Reference	100.0	100.0	0.909	0.585	0.708	0.734	–
EncDecA	21.57	37.64	0.862	**0.681**	0.648	0.730	n/a
RL w/ Zhang's Reward	23.30	**39.24**	**0.878**	0.659	**0.663**	0.734	1.91
RL w/ Zhao's Reward	**23.42**	39.20	**0.878**	0.662	0.662	0.734	1.69
RL w/ Our Reward	23.14	38.70	**0.878**	0.678	0.653	**0.736**	**1.45****

Table 3: Experimental results of text simplification. Note that G, M, S, and O correspond to grammaticality, meaning preservation, simplicity, and overall rewards, respectively. ** indicates a statistically significant difference between the others. (The p-value of the unpaired t-test of our method and both of the other methods were $p < 0.01$.)

Source	They are tired and it shows in their voices , but they 're still on the freedom highway .
Reference	Their voices sound tired .
EncDecA	They are tired and it shows in their voices , but they 're still on the freedom .
RL w/ Zhang's Reward	They are tired .
RL w/ Zhao's Reward	They are tired .
RL w/ Our Reward	They are tired and it shows in their voices .
Source	Historic architecture , crafts and music are being overwhelmed by China 's growth and its inability to effectively preserve traditions of the past .
Reference	Kite making is only part of a bigger story in China .
EncDecA	Historic architecture , crafts and music are being overwhelmed by China 's growth and its inability to effectively preserve traditions of the past .
RL w/ Zhang's Reward	Historic architecture , crafts and music are being overwhelmed by China 's growth .
RL w/ Zhao's Reward	The music of the city 's growth and music are being overwhelmed by China 's growth .
RL w/ Our Reward	Historic architecture , crafts and music are being overwhelmed by China 's growth and its actions .

Table 4: Examples of generated sentences by each simplification model.

both BLEU and SARI metrics. Reinforcement learning also improved rewards, but the EncDecA baseline was the best for meaning preservation. A trade-off relationship was observed between the rewards of meaning preservation and simplicity. This is expected because the meanings of the input and generated sentences deviate as the model replaces and deletes tokens for simplicity. Reinforcement learning based on our rewards does not improve simplicity as much as previous methods, but it does not worsen meaning preservation. This balance has made our model achieve the highest overall reward.

Table 4 shows generated sentences by each model. While the previous methods generates extremely simple sentence at the expense of meaning preservation, our model generates sentences with reasonable balances between meaning preservation and simplicity.

4.4 Results of Human Evaluation

We also conducted human evaluation using Amazon Mechanical Turk.[8] Human evaluators rank three sentences generated by a model based on reinforcement learning with different rewards, taking into account the source sentence. Three sentences were ranked on the basis of whether they were rewritten in a simple manner while preserving as much of the meaning of the source sentence as pos-

[8] https://www.mturk.com/

sible. We randomly selected 100 sets of generated sentences, excluding examples that all models generated the same sentences.[9] To ensure the quality of the human evaluation, we employed five master workers for each example and used 85 examples with at least three of them had the same ranking order.

The average ranking of each model is shown in the last column of Table 3. Our model was ranked significantly higher than previous models as confirmed by bootstrap testing. These results confirm that our rewards allow to generate simplified sentences preferred by humans.

5 Conclusion

We trained a text simplification model based on reinforcement learning with rewards that are highly correlated with human-perspectives. Experimental results showed that existing rewards employing evaluation metrics tend to generate extremely simple sentence at the expense of meaning preservation. Nevertheless, our BERT-based rewards succeeded in balancing meaning preservation and simplicity. In addition, we confirmed that human evaluators prefer our simplified sentences to those generated by previous rewards.

In this study, we set the equal weights to our sub-rewards. We plan to investigate the better weight balance of sub-rewards in the future.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP20K19861.

References

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A Dataset for Tuning and Evaluation of Sentence Simplification Models with Multiple Rewriting Transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *Proceedings of the 5th International Conference on Learning Representations*.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 269–270.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 1–14.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. In *Proceedings of the 28th Conference on Neural Information Processing Systems*, pages 3079–3087.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2019. Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3115–3125.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting Grammaticality on an Ordinal Scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 174–180.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text Simplification for Reading Assistance: A Project Note. In *Proceedings of the 2nd International Workshop on Paraphrasing*, pages 9–16.

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. *Technical report, Defence Technical Information Center (DTIC) Document*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, pages 74–81.

[9] We included examples that two models output the same sentences.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *Proceedings of the 6th International Conference on Learning Representations*.

Sarah E. Petersen and Mari Ostendorf. 2007. Text Simplification for Language Learners: A Corpus Analysis. In *Proceedings of the SLaTE Workshop on Speech and Language Technology in Education*, pages 69–72.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations*.

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*, pages 1715–1725.

Sanja Štajner and Maja Popović. 2016. Can Text Simplification Help Machine Translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Sanja Štajner and Maja Popović. 2018. Improving Machine Translation of English Relative Clauses with Automatic Text Simplification. In *Proceedings of the 1st Workshop on Automatic Text Adaptation*, pages 39–48.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. BLEU is Not Suitable for the Evaluation of Text Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744.

David Vickrey and Daphne Koller. 2008. Sentence Simplification for Semantic Role Labeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 344–352.

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence Simplification with Memory-Augmented Neural Networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 79–85.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771*.

Kristian Woodsend and Mirella Lapata. 2014. Text Rewriting Improves Semantic Role Labeling. *Journal of Artificial Intelligence Research*, 51:133–164.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A Study of Reinforcement Learning for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-Supervised Text Simplification with Back-Translation and Asymmetric Denoising Autoencoders. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 9668–9675.

Label Representations in Modeling Classification as Text Generation

Xinyi Chen[*]
New York University
xc1121@nyu.edu

Jingxian Xu[*]
New York University
jx880@nyu.edu

Alex Wang
New York University
alexwang@nyu.edu

Abstract

Several recent state-of-the-art transfer learning methods model classification tasks as text generation, where labels are represented as strings for the model to generate. We investigate the effect that the choice of strings used to represent labels has on how effectively the model learns the task. For four standard text classification tasks, we design a diverse set of possible string representations for labels, ranging from canonical label definitions to random strings. We experiment with T5 (Raffel et al., 2019) on these tasks, varying the label representations as well as the amount of training data. We find that, in the low data setting, label representation impacts task performance on some tasks, with task-related labels being most effective, but fails to have an impact on others. In the full data setting, our results are largely negative: Different label representations do not affect overall task performance.

1 Introduction

State-of-the-art transfer learning methods model classification tasks as text generation, such as GPT2 (Radford et al., 2019) and T5 (Raffel et al., 2019), and have led to significant improvements across a variety of NLP tasks. In this setting, labels are represented as strings for the model to generate, and the pretrained language model is finetuned to maximize the probability of generating the chosen label representation. For example, for CoLA (Warstadt et al., 2019) task, a classifier will be trained to output 0 (representing unacceptable) or 1 (representing acceptable), while T5 models it as a generation task, it is trained to generate the string "unacceptable" or "acceptable". The advantage of this approach is that the language model can be applied to the classification task as-is, without the need for any additional task-specific parameters or training.

However, in this setting, the impact of the particular strings used to represent labels remains unclear on the end task performance. One of the few studies on this question find that the linguistic properties (relatedness, polarity scale, etc.) of the labels do affect task performance (Nogueira et al., 2020), though their results are limited to document retrieval. We therefore further investigate the impact of string representation when modeling text classification as text generation.

We experiment with T5-base and four diverse, standard text classification tasks. For each task, we design a wide range of label representations, including canonical task labels; task-unrelated antonyms; and completely random strings. As previous works by Nogueira et al. (2020) have noted that the impact of label representation is particularly noticeable in lower data settings, we also vary the amount of training data for each task.

Our experiments reveal that, in the full data setting, the choice of label representation largely does not affect overall performance, with only one of the four datasets seeing any impact. In the low data setting, label representations sometimes have an impact on the overall performance, with task-related labels being the most effective in these cases.

2 Related Work

Target Word Probing Experiments Nogueira et al. (2020) probe the effects of label representation on document retrieval and ranking. They set the baseline mapping as {Positive → `true`, Negative → `false`}, and also try the reverse mapping, antonyms, related words, unrelated words and subwords. In the low data setting, they find that the baseline mapping yields accuracy significantly higher than other types of mappings do. In the high

[*] These two authors contributed equally.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 153–157

data setting, related words mapping is the most effective, though the differences between mappings are not as large as in the low-data regime. We extend these experiments with more diverse tasks and label representations.

Cloze Reformulation Schick and Schütze (2020) introduce Pattern Exploiting Training, where the input is transformed into a Cloze-style sentence. For example, the task of identifying whether two sentences a and b contradict or agree with each other is reformulated into "a? [blank], b.", and a pretrained language model is used as-is to generate `Yes` or `No` to fill in the blank. They claim that this procedure significantly improves the performance on several tasks in the zero-shot setting. Similarly, Petroni et al. (2019) probe the knowledge presented in state-of-the-art language models without fine-tuning using similar sentences with blanks. They find that these language models contain rich factual knowledge from pretraining, and are effective at recalling knowledge when answering fact-related questions.

Prompt Design Jiang et al. (2020) propose several methods to automatically generate efficient prompts for extracting knowledge from pretrained models, rather than manually designing them. They find that different templates, e.g. "x who converted to y", compared to "x is affiliated with y religion", can improve accuracy by as much as 60%. This line of work is orthogonal to ours: We focus on optimal label representation whereas they focus on the best way to format inputs.

3 Label Representations

We consider four standard text classification datasets representing different tasks and textual genres: (i) sentence acceptability judgments with CoLA (Warstadt et al., 2019); (ii) sentiment analysis with SST-2 (Socher et al., 2013); (iii) paraphrase detection with PAWS (Zhang et al., 2019); and (iv) commonsense reasoning with COPA (Roemmele et al., 2011). For each of these datasets, we test a wide variety of label representations.

3.1 Random Labels

As a simple baseline, we test whether the labels need to be semantic at all. We focus on random labels with short lengths, e.g. {`unacceptable` → `i`, `acceptable` → `c`}.

3.2 Task-Unrelated Labels

Next, we choose sets of words for labels based on their relationship to the task and to each other. We generate them by the following different rules:

- **Antonyms:** We choose words that are antonyms but are semantically unrelated to our tasks, e.g. {`unacceptable` → `cold`, `acceptable` →`hot`}. This setting tests whether it is important that labels be task related or if it is sufficient that they have opposing meanings.
- **Synonyms:** To contrast the antonyms, we use words that are synonyms but are semantically unrelated to our tasks, e.g. {`unacceptable` → `cold`, `acceptable` → `chilly`}.
- **Irrelevant words:** We choose words that are not related to the task nor each other, e.g. {`unacceptable` → `ice`, `acceptable` → `happy`}.
- **Relevant words:** We pick words that are relevant to each other but are not antonyms or synonyms, e.g. {`unacceptable` → `apple`, `acceptable`: → `orange`}.

3.3 Task-Related Labels

We further study how much performance varies between semantically similar task-related label representations. We choose sets of words as labels that have the same polarity scale or meanings as the original targets, e.g. {`unacceptable` → `no`, `acceptable` → `yes`}; As an additional baseline, we use labels that have the opposite polarity or meaning as the original labels, e.g. {`unacceptable` → `yes`, `acceptable` → `no`}.

4 Experiments and Results

4.1 Model and Optimization

Inspired by Nogueira et al. (2020), we experiment with T5 (Raffel et al., 2019), specifically the T5-base model. Among the four datasets, PAWS is the only dataset that has not been pretrained on by T5. So, for CoLA, SST-2, and COPA, we format each example the same as by T5. For PAWS, because it is the same task as MRPC (Dolan and Brockett, 2005), we format the examples identically to how Raffel et al. (2019) process MRPC. See Table 1 for examples.

(CoLA) **input**: `cola sentence: John made Bill master of himself.` **target**: `acceptable`
(SST-2) **input**: `sst2 sentence: it confirms fincher 's status as a film maker who artfully bends technical know-how to the service of psychological insight.` **target**: `positive`
(COPA) **input**: `copa choice1: Many citizens relocated to the capitol. choice2: Many citizens took refuge in other territories. premise: Political violence broke out in the nation. question: effect` **target**: `choice2`
(PAWS) **input**: `paws sentence1: In the early years , KETS was associated with National Educational Television , the forerunner of the current PBS. sentence2: In the early years , KETS was connected with National Educational Television , the forerunner of the current PBS.` **target**: `equivalent`

Table 1: Input formatting for all tasks, with the original canonical label representations from Raffel et al. (2019).

For COPA, CoLA, and SST-2, we finetune with optimize Adam (Kingma and Ba, 2014) for 2000 steps with learning rate 10^{-4}. For PAWS, we finetune for 3000 steps and similarly evaluate the accuracy of the development set every 200 steps. For all datasets, we evaluate the accuracy of the development set every 200 steps and report the best accuracy. For each dataset and setting, we evaluate three runs with different random seeds.

4.2 Low-Data Settings

As Nogueira et al. (2020) found significant differences in performance with distinct data sizes, we also run all four tasks in lower-data regimes. We choose the datasets mainly following these rules: For all tasks, we use the full training sets. Then, we downsample the training set to sizes for which we observe larger gaps between different labels than at the full dataset size. For each run with the low-data setting, we randomly generate a sample before training it, so the sample set normally varies for each run. To test an extremely low-data setting, for COPA and SST-2, we also choose a dataset with an absolute size of 10.

4.3 Results

We present results for SST-2 and COPA in Table 2 and for PAWS and CoLA in Table 3. In the full data setting, we obtain performance near that of BERT (Devlin et al., 2018) for all tasks. It is notable that even in the extremely low data settings, we obtain nontrivial performance for all tasks except COPA, which is likely due to the amount of pretraining in even the T5-base model.

For SST-2, by training on the full dataset, we get similar results for all choices of label representation. Notably, even the random strings perform as well as the task-specific labels. However, given the large size of the full dataset, it is unsurprising that the model can learn the class definitions from random strings.

For the 100 and 10 example settings, we find that the original labels achieve the best accuracy, 2% less than the full data setting. However, a potential confounder is the fact that T5 was pretrained on SST-2 with these labels, which likely explains the high accuracy with only 10 examples. We find the mutually unrelated and task-unrelated labels perform as well as the original labels in the 100 example setting (`ice`/`happy`), suggesting that even with this few examples, the choice of labels or even the relationship between them is not crucial. Similarly, in the 10 example setting, these mutually unrelated labels (`ice`/`happy`) perform as well as task-specific labels and task-unrelated antonyms, providing further evidence that the choice of labels is not crucial to learning the task, even in extremely low data settings. On the other hand, reversing the original labels or reversing task-related labels consistently performs the worst, even worse than random labels. This suggests that the label representations do not matter, as long as we do not pick labels that flip the class definition.

For COPA, in the full data setting, we observe notable performance differences between various labels. We get the best performance using the original labels and other task-related labels, while task-unrelated labels generally perform much worse. For the random labels, we observe high variability,

		SST-2			COPA		
label type / training set size		67349 (100%)	100 (0.167%)	10 (0.015%)	400 (100%)	80 (20%)	10 (2.5%)
original		94.48 ± 0.37	93.83 ± 0.29	93.92 ± 0.00	68.00 ± 2.00	65.00 ± 1.00	58.33 ± 0.58
reversed		94.07 ± 0.54	89.46 ± 1.00	31.54 ± 15.10	66.00 ± 3.61	63.67 ± 8.74	52.67 ± 5.13
random	i/c	94.38 ± 0.72	91.96 ± 0.90	72.36 ± 3.74	59.00 ± 5.00	54.67 ± 0.58	52.00 ± 4.00
	n/p	93.96 ± 0.07	91.67 ± 0.76	73.43 ± 3.46	68.33 ± 3.51	62.67 ± 2.52	54.67 ± 4.73
related	matched	94.50 ± 0.23	92.00 ± 0.87	75.96 ± 0.29	69.67 ± 2.31	68.33 ± 2.08	51.33 ± 9.24
	reversed	94.46 ± 0.24	90.50 ± 2.18	54.01 ± 0.53	69.33 ± 3.79	57.00 ± 6.00	43.67 ± 2.31
unrelated	cold/hot	94.75 ± 0.15	92.17 ± 0.76	84.29 ± 3.00	62.67 ± 2.52	55.00 ± 1.00	46.67 ± 2.08
	cold/chilly	94.19 ± 0.18	91.86 ± 0.92	73.05 ± 7.47	67.00 ± 2.65	61.33 ± 1.53	52.67 ± 1.53
	apple/orange	94.00 ± 0.18	91.17 ± 1.04	65.14 ± 2.56	55.67 ± 2.08	53.33 ± 2.08	52.00 ± 3.46
	ice/happy	94.69 ± 0.18	93.67 ± 0.58	84.29 ± 3.51	56.33 ± 4.04	54.33 ± 4.73	49.00 ± 3.00

Table 2: Accuracy (mean ± std) for SST-2 and COPA for a variety of dataset sizes and label representations. The original labels for SST-2 and COPA are 'negative/positive' and 'choice1/choice2', respectively. Matched labels for SST-2 and COPA are 'no/yes' and 'first/second', respectively.

		PAWS			CoLA		
label type / training set size		49401 (100%)	4000 (8.10%)	400 (0.81%)	8551 (100%)	4000 (46.8%)	400(4.68%)
original		94.09 ± 0.14	91.48 ± 0.24	83.50 ± 0.54	58.64 ± 0.15	57.15 ± 0.30	48.67 ± 0.00
reversed		94.68 ± 0.06	91.48 ± 0.24	85.04 ± 0.75	56.58 ± 1.95	55.78 ± 1.22	33.48 ± 6.74
random	i/c	93.00 ± 0.36	91.02 ± 0.26	82.50 ± 4.05	58.53 ± 1.28	56.70 ± 0.53	49.28 ± 3.16
	n/p	93.58 ± 0.44	91.49 ± 0.55	83.50 ± 1.82	57.14 ± 0.95	57.81 ± 1.75	46.85 ± 1.25
related	matched	93.63 ± 0.23	91.08 ± 0.45	84.83 ± 0.51	57.47 ± 2.43	55.59 ± 1.17	41.30 ± 4.23
	reversed	93.42 ± 0.51	92.81 ± 1.22	84.38 ± 1.19	57.22 ± 0.88	55.82 ± 0.39	28.33 ± 1.70
unrelated	cold/hot	93.12 ± 0.47	91.73 ± 0.53	83.50 ± 2.10	57.53 ± 0.52	56.18 ± 1.14	39.21 ± 2.39
	cold/chilly	93.55 ± 0.54	91.30 ± 0.19	84.28 ± 1.56	58.57 ± 3.01	57.47 ± 0.41	45.27 ± 1.58
	apple/orange	93.85 ± 0.21	91.29 ± 0.27	82.50 ± 0.99	58.77 ± 1.48	56.48 ± 1.27	33.38 ± 1.32
	ice/happy	93.16 ± 0.13	91.67 ± 0.38	83.00 ± 2.38	56.97 ± 1.80	56.34 ± 1.26	47.64 ± 2.67

Table 3: Accuracy (mean ± std) for PAWS and Matthews' correlation coefficient for CoLA for a variety of dataset sizes and label representations. The original labels for PAWS and CoLA are 'not_equivalent/equivalent' and 'unacceptable/acceptable', respectively. Matched labels for PAWS and CoLA are 'different/same' and 'no/yes', respectively.

with one pair of labels (n/p) worryingly performing as well as the original task labels. Also concerning is the fact that reversing the original or other task-related labels performs nearly as well as not reversing them. We speculate that the small dataset size contributes to these odd performance trends, as has been previously noted for COPA (Sap et al., 2019).

In the extremely low data settings, differences between labels become more significant and noticeable, with several label representations failing to learn the task and obtaining accuracy near chance (50%). The original and reversed pairs, as well as the task-related pairs, show similar accuracies, and these labels generally perform the best among all labels. Similar to SST-2, reversing the original labels performs worse and shows more instability. In the smallest setting, matched task-related labels also perform better than reversed ones and all other labels except for the original label pair.

For CoLA and PAWS, in full data and 4000 sample data regimes, the performances we get using different labels vary, but by no more than 2%. When the data size is extremely low, we observe notable gaps among labels for CoLA, but not for PAWS. Similar to SST-2, for CoLA, we find that reversed labels perform much worse than original ones.

5 Conclusion

In this work, we investigate the impact of label representation in modeling classification as a seq2seq task. For four standard text classification datasets and task types, we design a wide range of label representations, ranging from canonical task-related labels to task-unrelated antonyms to random words and strings. We experiment with the T5-base model on these datasets and label representations in a range of regimes with various data sizes.

Overall, we find that the choice of label representation largely does not affect task performance, though it varies by task and dataset size. In the high data settings, there is generally no differences between choices of label representations, and even random strings can function well as label repre-

sentations. In low data settings, the influence of label representations varies significantly between tasks. For PAWS, we observe no variation, but for COPA and CoLA, we note that task-related labels generally perform best.

Our experiments represent preliminary negative evidence that label representations have limited impact on task performance, but there are a number of dimensions that our work does not investigate that might affect a model's sensitivity to label representation.

First, we take as given that canonical task labels as sufficient for all tasks. However, for some tasks, the canonical label representation might be suboptimal or not sufficiently convey the semantics of the task, e.g. `choice1`/`choice2` for COPA. A natural way to mitigate this issue would be to use a much larger set of possible label representations, or even automatically discover label representations. However, we expect this issue will be especially problematic with more complex tasks that are difficult to represent with single-word labels or to succinctly represent in text at all.

Second, we use a single task input format, but the task formatting may be suboptimal and affect the ability of the model to learn from the label semantics. Schick and Schütze (2020) use multiple example templates per task and find that performance between templates can vary substantially. While an obvious direction would be to simply use multiple templates, automatically discovering effective templates also seems like a promising direction.

Acknowledgements

We thank Samuel R. Bowman, Nikita Nangia, and Marina Zavalina for their guidance and support; Yang Liu for his mentoring and advice.

References

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

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. *arXiv e-print 2003.06713*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv e-print 1910.10683*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *EMNLP*.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv e-print 2001.07676*.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling.

Generating Inflectional Errors for Grammatical Error Correction in Hindi

Ankur Sonawane, Sujeet Kumar Vishwakarma,
Bhavana Srivastava, Anil Kumar Singh
Indian Institute of Technology (BHU),Varanasi
{sankur.shrikant.eee16, sujeetkr.vishwakarma.eee16}@itbhu.ac.in
{bhavanasrivastava.rs.cse17, aksingh.cse}@iitbhu.ac.in

Abstract

Automated grammatical error correction has been explored as an important research problem within NLP, with the majority of the work being done on English and similar resource-rich languages. Grammar correction using neural networks is a data-heavy task, with the recent state of the art models requiring datasets with millions of annotated sentences for proper training. It is difficult to find such resources for Indic languages due to their relative lack of digitized content and complex morphology, compared to English. We address this problem by generating a large corpus of artificial inflectional errors for training GEC models. Moreover, to evaluate the performance of models trained on this dataset, we create a corpus of real Hindi errors extracted from Wikipedia edits. Analyzing this dataset with a modified version of the ERRANT error annotation toolkit, we find that inflectional errors are very common in this language. Finally, we produce the initial baseline results using state of the art methods developed for English.

1 Introduction

Grammatical Error Correction (GEC) involves automatically correcting errors in written text, whether relating to orthography, syntax or fluency. Today, most approaches for solving this problem highlight statistical and deep learning methods as opposed to rule-based methods. These methods treat GEC as a translation task, from an ungrammatical to a grammatically correct form of the same language (Brockett et al., 2006). This requires a considerable amount of supervised data in the form of 'edits', which are pairs of incorrect and correct sentences. Researchers have recently done remarkable work on English and a few other resource-rich languages and have released many datasets to evaluate state of the art methods. Comparatively less attention has been given to low resource languages, and Indic languages have been neglected in particular. Systems like UTTAM (Jain et al., 2018) and SCMIL (Etoori et al., 2018) have applied probabilistic approaches and deep learning, respectively, to the problem of spelling correction in Indic languages. Moreover, simple n-gram based models (Singh and Singh, 2019; Kanwar et al., 2017) have been used for "Real-Word" error correction, which is a very similar problem to GEC. However, to our knowledge, no such work exists for true GEC in this language. Thus, we sum up our contributions in the following manner:

1. We create a parallel corpus of synthetic errors by inserting errors into grammatically correct sentences using a rule-based process, focusing specifically on inflectional errors. Since this process is generic, it can easily be extended to other Indic languages.
2. We scrape Hindi edits from Wikipedia and filter them to provide another smaller corpus of errors. Since this corpus is extracted from a relatively natural source, it can be useful for evaluating GEC systems. We also analyze this corpus using an extended version of the ERRANT toolkit.
3. We evaluate a few well studied approaches for languages like English on these datasets, and thus produce the initial GEC results for the Hindi language. The code and data to reproduce our experiments are available at http://github.com/s-ankur/hindi_grammar_correction.

2 Related Work

The most common GEC datasets come from correction-annotated language learner essays. The English learner corpora include those from shared

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 158–164

tasks such as Helping Our Own (Dale et al., 2012), CoNLL2014 (Ng et al., 2014) and recently, BEA2019 (Bryant et al., 2019). Similar learner corpora exist for the Russian (Rozovskaya and Roth, 2019) and the Czech (Náplava and Straka, 2019) languages. However, the problem with such manually annotated corpora is that they are not readily available for low resource languages, and their creation will be resource and time-intensive.

Another popular method has been the deliberate injection of errors into grammatically correct sentences, whether by a rule-based system or by strategies like round-trip translation (Lichtarge et al., 2019). The former approach has been essential for languages with limited training data. This was the case for English early on (Izumi et al., 2004; Foster and Andersen, 2009), and is still the case for low resource languages such as Indonesian (Irmawati et al., 2017). Provided that the artificial errors closely resemble real-world mistakes, this method can be applied to obtain large volumes of training data reliably.

A third approach involves mining edits from websites, such as language learner websites (Mizumoto et al., 2011) or from websites with public revision histories like Wikipedia[1] (Grundkiewicz and Junczys-Dowmunt, 2014; Faruqui et al., 2018; Boyd, 2018) or GitHub (Hagiwara and Mita, 2020). While this has the potential to yield natural datasets of considerable size, there are several issues with edits obtained by this method, as not all corrections made in the text are of a grammatical nature; and many simply add more information or are semantic improvements to the text. As the edits lack any human curation, this method results in a more noisy corpus.

3 Hindi Grammar

Hindi is a fusional language that expresses grammatical features like case, gender, number, tense, etc. via morphological changes. In particular, all verbs and some adjectives are inflected to agree with the number and gender of the associated noun (Shapiro, 2003). The same is the case for genitive pronouns, genitive post-positions, and ordinals. Additionally, the verb inflects for the person and the adjective declines for the case of the noun. With a few exceptions, these changes are indicated by vowel endings to the right of the lexical base, as shown with examples in Table 1. If the proper inflection is not specified, then the sentence becomes easily identifiable as ungrammatical due to the loss of agreement.

Gender	Singular	Plural
Masculine	करता karatā	करते karate
Feminine	करती karatī	करती karatī

Table 1: Paradigm for the verb करना (karanā, *"to do"*), showcasing the change in endings according to the gender and number.

4 Error Extraction from Hindi Wikipedia

The WikiEdits 2.0[2] (Grundkiewicz and Junczys-Dowmunt, 2014) software uses Wikipedia revision histories to extract a parallel corpus of errors. We modify this tool for Hindi and extract edits from a Wikipedia revision dump dated October 1, 2020 to create our dataset, which we term as HiWikEd. For filtering the edits, we constrain extracted sentence length to between 6 and 27 tokens and consider only substitution operations with a token-based Levenstein edit distance of less than 0.3. Additionally, we discard edits containing only a difference in punctuation or numbers and corrections involving extremely rare tokens or HTML markups. Edits relating to vandalism are also discarded.

5 Error Analysis

The ERRor ANotation Toolkit (ERRANT[3]) (Bryant et al., 2017; Felice et al., 2016) is a tool that uses morphological and dependency information to analyze, merge and categorize errors using a rule-based system. Initially created for English, it has since been extended to German (Boyd, 2018). We use a similar method to extend the toolkit to Hindi and use it to classify the errors in HiWikEd (See Table 2 for examples). Although the classification criteria consider many exceptional cases, the basic reasoning used by us is as follows:

1. POS tags and lemma for the tokens are obtained using the StanfordNLP tagger (Qi et al., 2018). By comparing POS tags for the edit, the error category is decided as follows.

[1] via http://dumps.wikimedia.org/

[2] http://github.com/snukky/wikiedits

[3] http://github.com/chrisjbryant/errant

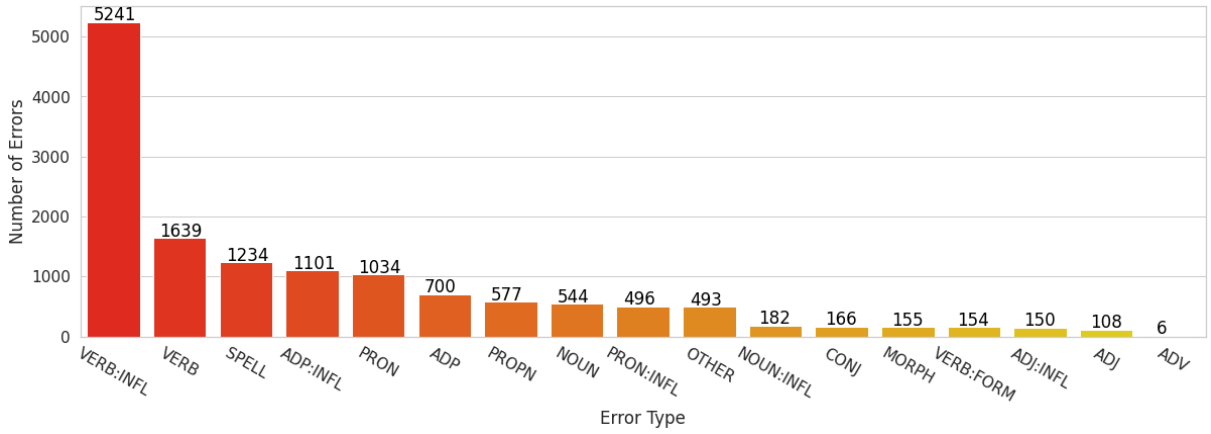


Figure 1: Frequencies of various error types in the HiWikEd dataset.

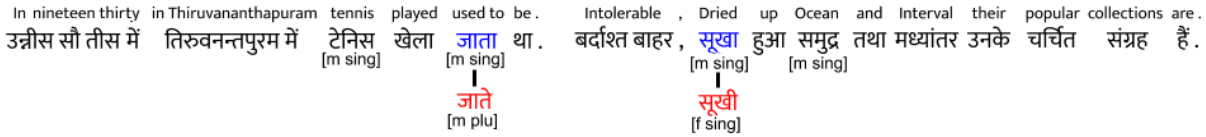


Figure 2: Example of error insertion. In the first sentence, the verb जाता (jātā, *"used to"*) agrees with the noun टेनिस (tenis, *"tennis"*) . We change the inflection of the verb and thus introduce disagreement into the sentence. The same is the case for the adjective सूखा (sūkhā, *"dry"*) in the next sentence.

2. Edits with the same lemma and POS are classified as <POS>:INFL errors and are grammatical in nature. For verbs, an additional category is introduced for tense termed as VERB:FORM.
3. Edits with different lemma but with the same POS are classified as <POS> errors. Most of these are simple semantic changes where one word is swapped for another (e.g. a synonym).
4. Edits having the same stem are classified as MORPH errors.
5. Edits with a low edit distance are classified as SPELL errors while the rest remain unclassified as OTHER.

6 Artificial Error Generation

Since inflectional errors form an easy to identify and common class of Hindi errors, we choose them to generate a synthetic dataset using the following process.

We first extract sentences from the Hindi Wikipedia revision dated June 1, 2020 (using WikiExtractor[4]), assuming that the recent versions are mostly grammatically correct. We tokenize these sentences and POS tag them using the Hindi POS Tagger (Reddy and Sharoff, 2011). We change the inflectional ending for all words of the VERB, ADP, ADV and PRON categories to a different random ending from the inflection table for that POS, taking care that exceptional cases are adequately handled (for examples, refer Table 3). For each of these changes, we create an edit containing a single incorrect word (See Figure 2). We randomly discard 40% of the sentence pairs thus generated. Keeping all sentences generated from a particular correct sentence in the same partition, we split the obtained dataset into train(80%) and valid(20%) partitions (refer Table 6).

7 Experiments and Evaluation

In our experiments, we first test the feasibility of the system using a basic transformer architecture (Vaswani et al., 2018) implemented using the Tensor2Tensor[5] library. For this we use the *transformer_base* setting as a baseline and train the model for 5K epochs. We then evaluate slightly modified versions of two state of the art models relating to English GEC. First, we train the multi-layer convolutional encoder-decoder model (Chollampatt and Ng, 2018)[6] for 5 epochs using the de-

[4] http://github.com/attardi/wikiextractor

[5] http://github.com/tensorflow/tensor2tensor

[6] http://github.com/nusnlp/mlconvgec2018

Error Type	Examples
VERB:FORM Verb Tense	बन(ban) → बना(banā), मिलते(milte) → मिलने(milne) make[pres → past], meet[past → inf]
VERB:INFL Verb Inflection	हुआ(huā)→ हुई(huī), रहता(rahatā) → रहते(rahate) happen[m.sing → f.sing], stay[m.sing → m.pl]
NOUN:INFL Noun Inflection	सदस्य(sadasya) → सदस्यों(sadasyon), जिले(jile) → जिला(jilā) member[nom → oblique], district[oblique → nom]
ADP:INFL Postposition Inflection	का(kā) → की(kī), का(kā) → के(ke) of[m.sing → f.sing], of[m.sing → pl]
PRON:INFL Pronoun Inflection	उसका(usakā) → उसकी(usakī), आपने(āpane) → आपको(āpako) his[m.sing → f.sing], you[erg → dat]
ADJ:INFL Adjective Inflection	छोटा(chhotā) → छोटे(chhote), दूसरे(dūsare) → दूसरा(dūsarā) small[m.sing → m.pl], other[m.sing.acc → m.sing]
VERB Verb	रखने(rakhane) → करने(karane), मिला(milā) → दिया(diyā) to keep → to do, found → gave
NOUN Noun	सदी(sadī) → शताब्दी(shatabdī), विश्वास(wishwāsa) → शासन(shāsan) century → centenary, trust → government
ADP Postposition	में(men) → , को(ko), से(se) → का(kā) in → to, from → of[m]
PRON Pronoun	उसके(usake) → उनके(unake), ये(ye) → आप(āp) his → their, these → you
ADJ Adjective	सामान्य(sāmānya) → आम(ām), बड़ा(badā) → छोटा(chhotā) common → ordinary, big → small
ADV Adverb	साथ(sāth) → बाद(bād), बराबर(barābar) → लगातार(lagātār) together → after, correctly → fast
CONJ Conjunction	अगर(agar) → यदि(yadi), पर(par) → परंतु(parantu) if → whether, but → however
MORPH Morphological	बनना(banana) → बनाने(banāne) become → make
SPELL Spelling	कौवा(kauwā) → कौआ (kauā), गई(gaī) → गयी(gayī) crow[spelling], went[spelling]
OTHER Unclassified	और (aur) → के(ke), शहर(shahar) → भी(bhī) and → of[pl], city → and also

Table 2: Error categories of HiWikEd as classified using ERRANT and examples (original → edited).

Error Type	Examples
ADP:INFL Postposition Inflection	की(kī) → के(ke) of[f.sing → pl]
PRON:INFL Pronoun Inflection	मेरा(merā) → मेरी(merī), अपने(apane) → अपनी(apanī) my[m.sing → f.sing], our[m.pl → f.sing]
ADJ:INFL Adjective Inflection	लंबे(lambē) → लंबा(lambā), चौथा(chauthā) → चौथे(chauthē) long[m.pl → m.sing], fourth[m.sing → m.pl]
VERB:INFL Verb Inflection	करता(karatā) → करती(karatī), किये(kiyē) → की(kī) do[m.sing → f.sing], do[pl.past → f.past]

Table 3: List of word categories corrupted using our approach along with examples.

System	ADP:INFL		PRON:INFL		ADJ:INFL		VERB:INFL		Full dataset	
	$F_{0.5}$	GLEU	$F_{0.5}$	GLEU	$F_{0.5}$	GLEU	$F_{0.5}$	GLEU	$F_{0.5}$	GLEU
Transf	0.30	0.62	0.06	0.67	0.06	0.57	0.55	0.79	0.31	0.69
MLConv	0.66	0.81	0.26	0.86	0.36	0.83	0.65	0.83	0.35	0.73
CopyAug	0.70	0.84	0.29	0.71	0.39	0.69	0.70	0.87	0.49	0.80

Table 4: Results for the systems trained on the synthetic corpus and tested on the HiWikEd corpus including the $F_{0.5}$ and GLEU scores. We also specifically report the metrics for the four inflectional categories that we train on.

Source	इस पर उनके पिता राजा नतमस्तक हो गया (gayā). *"on this, his father, the king[m.pl], bowed[m.sing] down"*
Reference	इस पर उनके पिता राजा नतमस्तक हो गये (gaye). *"on this, his father, the king[m.pl], bowed[m.pl] down"*
Output_{Transf}	इस पर उनके पिता राजा नतमस्तक हो गया (gayā). *"on this, his father, the king[m.pl], bowed[m.sing] down"*
Output_{MLConv}	इस पर उनके पिता राजा नतमस्तक हो गये (gaye). *"on this, his father, the king[m.pl], bowed[m.pl] down"*
$\text{Output}_{CopyAug}$	इस पर उनके पिता राजा नतमस्तक हो गये (gaye). *"on this, his father, the king[m.pl], bowed[m.pl] down"*

Table 5: Example system outputs along with source and reference sentences from HiWikEd. The source sentence comes from the Etymology section of Wikipedia article for the Amer Fort.

Dataset	#Sent	#Tok	%Err
Synthetic (Train)	2.6M	45.5M	5.7
Synthetic (Valid)	0.5M	9.1M	5.7
HiWikEd (Test)	13K	208K	6.7

Table 6: Corpus statistics including error percentages, and number of sentences and tokens.

fault hyperparameters. Finally, for training the copy augmented transformer model (Zhao et al., 2019)[7], we skip the pretraining step with the denoising auto-encoder and train the system for 9 epochs.

For model evaluation, we use the GLEU metric (Napoles et al., 2015) as well as the $F_{0.5}$ metric calculated using the Max-Match(M^2) scorer[8](Dahlmeier and Ng, 2012). The systems were trained on the synthetic dataset and then evaluated on the HiWikEd corpus, and the results are presented in Table 4 with an example output shown in Table 5. In addition to the metrics on the full HiWikEd dataset, we calculate the per error type metrics by categorizing the edits using ERRANT, for a more fine-grained analysis of the results.

[7] http://github.com/yuantiku/fairseq-gec
[8] http://github.com/nusnlp/m2scorer

8 Discussion and Future Work

Motivated by the lack of work in GEC for Indic languages, we present two novel error corpora in the Hindi language (as shown in Table 6) and also provide a method for generating a large quantity of artificial inflectional errors. Following error analysis of the HiWikEd corpus using the ERRANT toolkit, we observe that inflectional errors are a reasonably common category in Hindi, making up 49.92% of all errors (see Figure 1).

As seen from the example outputs in Table 5 as well as from the metrics presented in Table 4, the models are able to properly correct many inflectional errors. As expected, the simpler Transformer model is significantly outperformed by the other two models. However, all of the methods perform relatively poorly with regard to the whole dataset, which contains numerous spelling errors and semantic edits for which we do not train our models.

In addition, some grammatical errors in HiWikEd were not inflectional (such as ADJ:FORM) and thus not represented in the synthetic dataset. On manual observation of the dataset, we also find that some edits are identifiably incorrect or simply denote stylistic differences and are out of the scope of GEC. Thus, it may be fruitful to filter and annotate the dataset manually. Including other error

types in the training dataset will undoubtedly improve the performance of the model.

Finally, while scraping edits from Wikipedia, we encountered numerous Hindi spelling errors, which we discarded as our focus was solely on grammatical errors. However, these edits may prove to be a valuable source of natural Hindi spelling errors, which can be used to circumvent the dataset problems faced by Etoori et al. (2018) and similar research. Since the approaches used by us for error generation and error categorization are not specific to Hindi, they can easily be extended to other Indic languages like Marathi and Bengali.

References

Adriane Boyd. 2018. Using Wikipedia edits in low resource grammatical error correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84, Brussels, Belgium. Association for Computational Linguistics.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada. Association for Computational Linguistics.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152, Melbourne, Australia. Association for Computational Linguistics.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A multilingual corpus of Wikipedia edits for modeling language and discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 305–315, Brussels, Belgium. Association for Computational Linguistics.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan. The COLING 2016 Organizing Committee.

Jennifer Foster and Oistein Andersen. 2009. GenERRate: Generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–90, Boulder, Colorado. Association for Computational Linguistics.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing*, pages 478–490, Cham. Springer International Publishing.

Masato Hagiwara and Masato Mita. 2020. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6761–6768, Marseille, France. European Language Resources Association.

Budi Irmawati, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Generating artificial error data for indonesian preposition error corrections. *International Journal of Technology*, 8(3):549–558.

Emi Izumi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. Sst speech corpus of japanese learners' english and automatic detection of learners' errors. *ICAME Journal*, 28:31–48.

Amita Jain, Minni Jain, Goonjan Jain, and Devendra K Tayal. 2018. "uttam" an efficient spelling correction system for hindi language based on supervised

learning. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(1):1–26.

Shailza Kanwar, Manoj Sachan, and Gurpreet Singh. 2017. N-grams solution for error detection and correction in hindi language. *International Journal of Advanced Research in Computer Science*, 8(7).

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356, Hong Kong, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Siva Reddy and Serge Sharoff. 2011. Cross language pos taggers (and other tools) for indian languages: An experiment with kannada using telugu resources. In *Proceedings of the Fifth International Workshop On Cross Lingual Information Access*, pages 11–19, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically rich languages: The case of Russian. *Transactions of the Association for Computational Linguistics*, 7:1–17.

Michael C Shapiro. 2003. Hindi. *The Indo-Aryan Languages*, pages 250–285.

S. Singh and S. Singh. 2019. Handling real-word errors of hindi language using n-gram and confusion set. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 433–438.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Author Index