EACL 2017

15th Conference of the European Chapter of the Association for Computational Linguistics

Proceedings of the Software Demonstrations

April 3-7, 2017
Valencia, Spain

Edited by
Anselmo Peñas
Universidad Nacional de Educacion a Distancia
Spain

Andre Martins
Unbabel and Instituto de Telecomunicacoes
Portugal

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 978-1-945626-36-4

Introduction

This volume contains the contributions accepted for Software Demonstrations at the EACL 2017 conference which takes place from 4th to 7th of April 2017 in Valencia, Spain.

The Demonstrations program primary aim is to present the software systems and solutions related to all areas of computational linguistics and natural language processing. This program encourages the early exhibition of research prototypes, but also includes interesting mature systems.

The number of submissions for software demonstrations exceeded, again, the number of available slots. This confronted us with the situation where we were forced to introduce a strict selection process. The selection process was completed following the predefined criteria of relevance for EACL 2017 conference: novelty, overall usability, and applicability to more than one linguistic level and to more than one computational linguistic field/task. We would like to thank the general chairs and the local organizers, without whom it would have been impossible to put together such a strong demonstrations program.

We hope you will find this proceedings useful and that the software systems presented in this proceedings will inspire new ideas and approaches in your future work.

With kind regards,

Demo chairs Anselmo Peñas and André Martins

General Chair (GC):

Mirella Lapata (University of Edinburgh)

Co-Chairs (PCs):

Phil Blunsom (University of Oxford)
Alexander Koller (University of Potsdam)

Local organizing team:

Paolo Rosso (Chair), PRHLT, Universitat Politècnica de València
Francisco Casacuberta (Co-chair), PRHLT, Universitat Politècnica de València
Jon Ander Gómez (Co-chair), PRHLT, Universitat Politècnica de València

Software Demo chairs

Anselmo Peñas (UNED, Spain)
Andre Martins (Unbabel and Instituto de Telecomunicacoes, Portugal)

Publication Chairs:

Maria Liakata
Chris Biemann

Workshop Chairs:

Laura Rimell
Richard Johansson

Student research workshop faculty advisor:

Barbara Plank

Tutorials co-chairs:

Alex Klementiev
Lucia Specia

Publicity Chair:

David Weir

Conference Handbook Chair:

Andreas Vlachos

Table of Contents

COVER: Covering the Semantically Tractable Questions
Michael Minock 1

Common Round: Application of Language Technologies to Large-Scale Web Debates
Hans Uszkoreit, Aleksandra Gabryszak, Leonhard Hennig, Jörg Steffen, Renlong Ai, Stephan Busemann, Jon Dehdari, Josef van Genabith, Georg Heigold, Nils Rethmeier, Raphael Rubino, Sven Schmeier, Philippe Thomas, He Wang and Feiyu Xu 5

A Web-Based Interactive Tool for Creating, Inspecting, Editing, and Publishing Etymological Datasets
Johann-Mattis List 9

WAT-SL: A Customizable Web Annotation Tool for Segment Labeling
Johannes Kiesel, Henning Wachsmuth, Khalid Al Khatib and Benno Stein 13

TextImager as a Generic Interface to R
Tolga Uslu, Wahed Hemati, Alexander Mehler and Daniel Baumartz 17

GraWiTas: a Grammar-based Wikipedia Talk Page Parser
Benjamin Cabrera, Laura Steinert and Björn Ross 21

TWINE: A real-time system for TWeet analysis via INformation Extraction
Debora Nozza, Fausto Ristagno, Matteo Palmonari, Elisabetta Fersini, Pikakshi Manchanda and Enza Messina 25

Alto: Rapid Prototyping for Parsing and Translation
Johannes Gontrum, Jonas Groschwitz, Alexander Koller and Christoph Teichmann 29

CASSANDRA: A multipurpose configurable voice-enabled human-computer-interface
Tiberiu Boroş, Stefan Daniel Dumitrescu and Sonia Pipa 33

An Extensible Framework for Verification of Numerical Claims
James Thorne and Andreas Vlachos 37

ADoCS: Automatic Designer of Conference Schedules
Diego Fernando Vallejo Huanga, Paulina Adriana Morillo Alcívar and Cesar Ferri Ramírez 41

A Web Interface for Diachronic Semantic Search in Spanish
Pablo Gamallo, Iván Rodríguez-Torres and Marcos Garcia 45

Multilingual CALL Framework for Automatic Language Exercise Generation from Free Text
Naiara Perez and Montse Cuadros 49

Audience Segmentation in Social Media
Verena Henrich and Alexander Lang 53

The arText prototype: An automatic system for writing specialized texts
Iria da Cunha, M. Amor Montané and Luis Hysa 57

QCRI Live Speech Translation System
Fahim Dalvi, Yifan Zhang, Sameer Khurana, Nadir Durrani, Hassan Sajjad, Ahmed Abdelali, Hamdy Mubarak, Ahmed Ali and Stephan Vogel 61

Nematus: a Toolkit for Neural Machine Translation
Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry and Maria Nadejde 65

A tool for extracting sense-disambiguated example sentences through user feedback
Beto Boullosa, Richard Eckart de Castilho, Alexander Geyken, Lothar Lemnitzer and Iryna Gurevych 69

Lingmotif: Sentiment Analysis for the Digital Humanities
Antonio Moreno-Ortiz 73

RAMBLE ON: Tracing Movements of Popular Historical Figures
Stefano Menini, Rachele Sprugnoli, Giovanni Moretti, Enrico Bignotti, Sara Tonelli and Bruno Lepri 77

Autobank: a semi-automatic annotation tool for developing deep Minimalist Grammar treebanks
John Torr 81

Chatbot with a Discourse Structure-Driven Dialogue Management
Boris Galitsky and Dmitry Ilvovsky 87

Marine Variable Linker: Exploring Relations between Changing Variables in Marine Science Literature
Erwin Marsi, Pinar Pinar Øzturk and Murat V. Ardelan 91

Neoveille, a Web Platform for Neologism Tracking
Emmanuel Cartier 95

Building Web-Interfaces for Vector Semantic Models with the WebVectors Toolkit
Andrey Kutuzov and Elizaveta Kuzmenko 99

InToEventS: An Interactive Toolkit for Discovering and Building Event Schemas
Germán Ferrero, Audi Primadhanty and Ariadna Quattoni 104

ICE: Idiom and Collocation Extractor for Research and Education
vasanthi vuppuluri, Shahryar Baki, An Nguyen and Rakesh Verma 108

Bib2vec: Embedding-based Search System for Bibliographic Information
Takuma Yoneda, Koki Mori, Makoto Miwa and Yutaka Sasaki 112

The SUMMA Platform Prototype
Renars Liepins, Ulrich Germann, Guntis Barzdins, Alexandra Birch, Steve Renals, Susanne Weber, Peggy van der Kreeft, Herve Bourlard, João Prieto, Ondrej Klejch, Peter Bell, Alexandros Lazaridis, Alfonso Mendes, Sebastian Riedel, Mariana S. C. Almeida, Pedro Balage, Shay B. Cohen, Tomasz Dwojak, Philip N. Garner, Andreas Giefer, Marcin Junczys-Dowmunt, Hina Imran, David Nogueira, Ahmed Ali, Sebastião Miranda, Andrei Popescu-Belis, Lesly Miculicich Werlen, Nikos Papasarantopoulos, Abiola Obamuyide, Clive Jones, Fahim Dalvi, Andreas Vlachos, Yang Wang, Sibo Tong, Rico Sennrich, Nikolaos Pappas, Shashi Narayan, Marco Damonte, Nadir Durrani, Sameer Khurana, Ahmed Abdelali, Hassan Sajjad, Stephan Vogel, David Sheppey, Chris Hernon and Jeff Mitchell 116

EACL 2017 Software Demonstrations Program

Wednesday 5th April 2017

17.30–19.30 Session 1

COVER: Covering the Semantically Tractable Questions
Michael Minock

Common Round: Application of Language Technologies to Large-Scale Web Debates
Hans Uszkoreit, Aleksandra Gabryszak, Leonhard Hennig, Jörg Steffen, Renlong Ai, Stephan Busemann, Jon Dehdari, Josef van Genabith, Georg Heigold, Nils Rethmeier, Raphael Rubino, Sven Schmeier, Philippe Thomas, He Wang and Feiyu Xu

A Web-Based Interactive Tool for Creating, Inspecting, Editing, and Publishing Etymological Datasets
Johann-Mattis List

WAT-SL: A Customizable Web Annotation Tool for Segment Labeling
Johannes Kiesel, Henning Wachsmuth, Khalid Al Khatib and Benno Stein

TextImager as a Generic Interface to R
Tolga Uslu, Wahed Hemati, Alexander Mehler and Daniel Baumartz

GraWiTas: a Grammar-based Wikipedia Talk Page Parser
Benjamin Cabrera, Laura Steinert and Björn Ross

TWINE: A real-time system for TWeet analysis via INformation Extraction
Debora Nozza, Fausto Ristagno, Matteo Palmonari, Elisabetta Fersini, Pikakshi Manchanda and Enza Messina

Alto: Rapid Prototyping for Parsing and Translation
Johannes Gontrum, Jonas Groschwitz, Alexander Koller and Christoph Teichmann

CASSANDRA: A multipurpose configurable voice-enabled human-computer-interface
Tiberiu Boroş, Stefan Daniel Dumitrescu and Sonia Pipa

An Extensible Framework for Verification of Numerical Claims
James Thorne and Andreas Vlachos

ADoCS: Automatic Designer of Conference Schedules
Diego Fernando Vallejo Huanga, Paulina Adriana Morillo Alcívar and Cesar Ferri Ramírez

Wednesday 5th April 2017 (continued)

A Web Interface for Diachronic Semantic Search in Spanish
Pablo Gamallo, Iván Rodríguez-Torres and Marcos Garcia

Multilingual CALL Framework for Automatic Language Exercise Generation from Free Text
Naiara Perez and Montse Cuadros

Audience Segmentation in Social Media
Verena Henrich and Alexander Lang

Thursday 6th April 2017

17.30–19.30 Session 2

The arText prototype: An automatic system for writing specialized texts
Iria da Cunha, M. Amor Montané and Luis Hysa

QCRI Live Speech Translation System
Fahim Dalvi, Yifan Zhang, Sameer Khurana, Nadir Durrani, Hassan Sajjad, Ahmed Abdelali, Hamdy Mubarak, Ahmed Ali and Stephan Vogel

Nematus: a Toolkit for Neural Machine Translation
Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry and Maria Nadejde

A tool for extracting sense-disambiguated example sentences through user feedback
Beto Boullosa, Richard Eckart de Castilho, Alexander Geyken, Lothar Lemnitzer and Iryna Gurevych

Lingmotif: Sentiment Analysis for the Digital Humanities
Antonio Moreno-Ortiz

RAMBLE ON: Tracing Movements of Popular Historical Figures
Stefano Menini, Rachele Sprugnoli, Giovanni Moretti, Enrico Bignotti, Sara Tonelli and Bruno Lepri

Autobank: a semi-automatic annotation tool for developing deep Minimalist Grammar treebanks
John Torr

Chatbot with a Discourse Structure-Driven Dialogue Management
Boris Galitsky and Dmitry Ilvovsky

Thursday 6th April 2017 (continued)

Marine Variable Linker: Exploring Relations between Changing Variables in Marine Science Literature
Erwin Marsi, Pinar Pinar Øzturk and Murat V. Ardelan

Neoveille, a Web Platform for Neologism Tracking
Emmanuel Cartier

Building Web-Interfaces for Vector Semantic Models with the WebVectors Toolkit
Andrey Kutuzov and Elizaveta Kuzmenko

InToEventS: An Interactive Toolkit for Discovering and Building Event Schemas
Germán Ferrero, Audi Primadhanty and Ariadna Quattoni

ICE: Idiom and Collocation Extractor for Research and Education
vasanthi vuppuluri, Shahryar Baki, An Nguyen and Rakesh Verma

Bib2vec: Embedding-based Search System for Bibliographic Information
Takuma Yoneda, Koki Mori, Makoto Miwa and Yutaka Sasaki

The SUMMA Platform Prototype
Renars Liepins, Ulrich Germann, Guntis Barzdins, Alexandra Birch, Steve Renals, Susanne Weber, Peggy van der Kreeft, Herve Bourlard, João Prieto, Ondrej Klejch, Peter Bell, Alexandros Lazaridis, Alfonso Mendes, Sebastian Riedel, Mariana S. C. Almeida, Pedro Balage, Shay B. Cohen, Tomasz Dwojak, Philip N. Garner, Andreas Giefer, Marcin Junczys-Dowmunt, Hina Imran, David Nogueira, Ahmed Ali, Sebastião Miranda, Andrei Popescu-Belis, Lesly Miculicich Werlen, Nikos Papasarantopoulos, Abiola Obamuyide, Clive Jones, Fahim Dalvi, Andreas Vlachos, Yang Wang, Sibo Tong, Rico Sennrich, Nikolaos Pappas, Shashi Narayan, Marco Damonte, Nadir Durrani, Sameer Khurana, Ahmed Abdelali, Hassan Sajjad, Stephan Vogel, David Sheppey, Chris Hernon and Jeff Mitchell

COVER: Covering the Semantically Tractable Questions

Michael Minock
KTH Royal Institute of Technology, Stockholm, Sweden
Umeå University, Umeå Sweden.
minock@kth.se, mjm@cs.umu.se

Abstract

In semantic parsing, natural language questions map to meaning representation language (MRL) expressions over some fixed vocabulary of predicates. To do this reliably, one must guarantee that for a wide class of natural language questions (the so called *semantically tractable questions*), correct interpretations are always in the mapped set of possibilities. Here we demonstrate the system COVER which significantly clarifies, revises and extends the notion of semantic tractability. COVER is written in Python and uses NLTK.

1 Introduction

The twentieth century attempts to build and evaluate natural language interfaces to databases are covered, more or less, in (Androutsopoulos et al., 1995; Copestake and Jones, 1990). While recent work has focused on learning approaches, there are less costly alternatives based on only lightly naming database elements (e.g. relations, attributes, values) and reducing question interpretation to graph match (Zhang et al., 1999; Popescu et al., 2003). In (Popescu et al., 2003), the notion of *semantically tractable questions* was introduced and further developed in (Popescu et al., 2004). The semantically tractable questions are those for which, under strong assumptions, one can guarantee generating a correct interpretation (among others). A focus of the PRECISE work was reducing the problem of mapping tokenized user questions to database elements to a max-flow problem. These ideas were implemented and evaluated in the PRECISE system, which scored 77% coverage over the full GEOQUERY corpus.

Here we demonstrate the system COVER, which extends and refines the basic model as follows: a) we explicitly describe the handling of self-joining queries, aggregate operators, sub-queries, etc.; b) we employ theorem proving for consolidating equivalent queries and including only queries that are information bearing and non-redundant; c) we more cleanly factor the model to better isolate the role of off-the-shelf syntactic parsers. In practice, our extended model leads to improved performance (12% absolute coverage improvement) on the full GEOQUERY corpus.

2 COVER

Figure 1 shows the components of COVER. The only configuration is a simple domain specific lexicon which itself may be automatically derived from the database instance and lexical resources. There are three core processing phases, which generate a set of MRL expressions from the user's question and two auxiliary processes which use off-the-shelf syntactic analysis and theorem proving to prune MRL expressions from the constructed set. We cover the three core phases here and the two auxiliary processes in section 3. For more detail see (Minock, 2017).

2.1 Phase 1: Generating All Mappings

A *mapping* is defined as an *assignment* of all word positions in a user's question to database elements (relations, attributes, values). Thus a mapping for the six-word question "what are the cities in Ohio?" would consist of six assignments. Since eventually mappings might generate MRL expressions, each assignment is marked by a type to indicate its use as either a *focus*, a *condition* or a *stop* assignment (to be ignored).

In COVER's first phase candidate mappings for a question are generated. For example in the question "what are the cities in Ohio?", under the correct mapping: 'what' is assigned to `City` and is marked as a focus type; 'are' and 'the' are marked

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 1–4

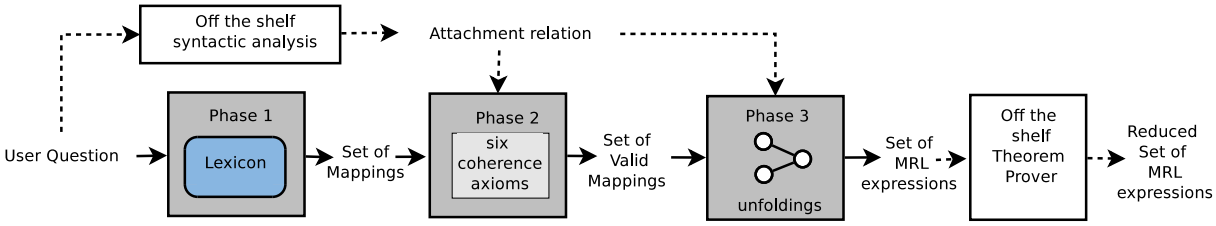


Figure 1: Overall processing paths of COVER.

as stop assignments and are assigned to the empty element; 'cities' is assigned to the relation `City` and is marked as condition type; 'in' is assigned the foreign key attribute `state` from the relation `City` to the relation `State` and is marked as a condition type; 'Ohio' is assigned to the value `'Ohio'` of attribute `name` of the relation `State` and is marked as a condition type. The knowledge to determine such mappings comes from a very simple domain lexicon which assigns phrases to database elements. (For example 'Ohio' in the lexicon is assigned to the database element representing the `State.name='Ohio'`). Note matching is based on word stems. Thus, for example, 'cities' matches 'city'.

2.2 Phase 2: Determining Valid Mappings

Valid mappings must observe certain constraints. For example the question "what is the capital of New York?" is valid when 'New York' maps to the state, but is not valid when 'New York' maps to the city. "What is the population of the Ohio River?" has no valid mappings because under the given database there is no way to fit the `population` attribute with the `River` table. Specifically the following are six properties that valid mappings must observe.

1. **There exists a unique focus element**. There always exists a unique focus element that is the attribute or relation upon which the question is seeking information. For efficiency this is actually enforced in phase 1.

2. **All relation focus markers are on mentioned relations**. If a relation is a focus, then it must be *mentioned*. A relation is mentioned if either it is explicitly named in the question (e.g. 'cities') or if a primary value[1] of the relation (e.g. 'New York' for the relation `City`) is within the question.

3. **All attribute focus markers are on mentioned, rooted attributes**. An attribute focus marker (e.g. 'what') must not only explicitly match an attribute (e.g. 'population'), but such an attribute must also be *rooted*. An attribute is rooted if the relation of the attribute is mentioned.

4. **Non-focus attributes satisfy correspondences**. Unless they are the focus, attributes must pair with a value (e.g. in "cities with *name Springfield*"), or, in the case that the attribute is a foreign key (e.g. "cities *in* the *state*..", the attribute must pair with the relation or primary value of the foreign key (e.g. "cities *in Ohio*").

5. **Value elements satisfy correspondences**. Values are either primary (e.g. "New York") or must be paired with either an attribute (e.g. "... city with the *name* of *New York*" ..."), or via ellipsis paired with a relation (e.g. "... the *city New York*").

6. **All mentioned elements are connected**. The elements assigned by the mapping must form a connected graph over the underlying database schema.

This leads to the core definition of this work:

Definition 1 *(Semantically Tractable Question) For a given question q, lexicon $\mathcal{L}$, q is semantically tractable if there exists at least one valid mapping over q.*

2.3 Phase 3: Generating Logical Formulas

Given a set of valid mappings, COVER's third phase is to generate one or more MRL expressions for each valid mapping. To achieve this we unfold the connected graph of valid mappings (see property 6 in section 2.2) into meaningful full

[1]Primary values(Li and Jagadish, 2014), stand, or for the most part stand, for a tuple of a relation. For example 'Springfield' is a primary value of City even though it is not a key value.

graphs. This is complicated in the self-joining case where graphs will have multiple vertices for a given relation. For example the valid mapping for "what states border states that border New York?" maps to two database relations: `State` and `Borders`. But the corresponding unfolded graph will be over three instantiations of `State` and two of `Borders`. Our algorithm gives a systematic and exhaustive process to compute all reasonable unfolded graphs. And then for each unfolded graph we generate the set of possible attachments of conditions and selections. In this end this gives a set of query objects which maybe directly expressed in SQL for application over the database or to first order logic expressions for satisfiability testing via an automatic theorem prover.

3 Auxiliary Processes

Figure 1 shows two auxiliary processes that further constrain what are valid mappings as well as which MRL expressions are returned.

3.1 Integrating Syntax

COVER uses an off-the-shelf parser to generate a syntactic attachment relation between word positions. This attachment relation is then used to sharpen the conditions in the properties 2, 3, 4 and 5 of valid mappings. In short correspondences and focus-value matches must be over word positions that are attached in the underlying syntactic parse. This has the effect of reducing the number of valid mappings. For example let us consider "what is the largest city in the smallest state?". If our syntactic analysis component correctly determines that 'largest' does not attach to 'state' and 'smallest' does not attach to 'city', then an erroneous second valid mapping will be excluded. Attachment information is also used to constrain which MRL expressions can be generated from valid mappings.

3.2 Integrating Semantics and Pragmatics

Many of the MRL expressions generated in phase three are semantically equivalent, but syntactically distinct. The second auxiliary process uses a theorem prover to reduce the number of MRL expressions (the shortest one from each equivalence class) that need to be paraphrased for interactive disambiguation. This is achieved by testing pairwise query containment over all the MRL expressions in the MRL set produced in the third phase.

Case	Coverage	Avg/Med # interpretations
full	780/880 (89%)	7.59/2
no-equiv reduction	780/880 (89%)	19.65/2

Table 1: Evaluation over GEOQUERY

Theorem proving is also used to enforce pragmatic constraints. For example we remove queries that do not bear information, or have redundancies within them that violate Gricean principles. This is principally achieved by determining whether a set-fetching query necessarily generates a single or no tuple where the answer is already in the question. For example a query retrieving "the names of states where the name of the state is New York and the state borders a state" does not bear information. Finally it should be noted, one can add arbitrary domain rules (e.g. *states have one and only one capital*) to constrain deduction. This would allow more query equivalencies and cases of pragmatics violations to be recognized.

4 Demonstration

Our demonstration, like PRECISE, is over GEOQUERY, a database on US geography along with 880 natural language questions paired with corresponding logical formulas. The evaluation method is exactly as in PRECISE(in conversation with Ana-Maria Popescu). First we prepare a lexicon over the GEOQUERY domain, then, given the set of 880 natural language/meaning representation pairs, the queries are run through the system and if the question is semantically tractable and generates one or more formal query expressions, then an expression equivalent to the target MRL expression must be within the generated set. Our experiment shows this to be the case.

Table 1 presents some results. By 'coverage' we mean, like in the PRECISE evaluation, that the answer is in the computed result set. Table 1 shows results for two cases: *full* has all features turned on; *no-equiv reduction* shows results when the auxiliary process described in section 3.2 is disengaged. Clearly we are benefiting from the use of a theorem prover which is used to reduce the size of returned query sets.

At EACL we will run our evaluation on a laptop, showing the complete configuration over GEOQUERY and welcoming audience members to pose interactive questions.

5 Discussion

To apply `Cover`, consider our strong, though achievable assumptions: a) *the user's mental model matches the database*; b) *no word or phrase of a question maps only to incorrect element, type pairs*; c) *all words in the question are syntactically attached*. Under such assumptions, every question that we answer is semantically tractable, and thus a correct interpretation is always included within any non-empty answer. Let us discuss the feasibility of these assumptions.

With respect to **assumption a**, it is difficult to determine if a user's mental model matches a database, but, in general, natural language interfaces do better when the schema is based on a conceptual (e.g. Entity-Relationship) or domain model, as is the case for GEOQUERY. Still COVER is not yet fully generalized for EER-based databases (e.g. multi-attribute keys, isa hierarchies, union types, etc.). We shall study more corpora (e.g. QALD (Walter et al., 2012)) to see what type of conceptual models are required.

With respect to **assumption b**, an over-expanded lexicon can lead to spurious interpretations, but that is not so serious as long as the right interpretation still gets generated. Any number of additional noisy entries could be added and our strong assumption b) would remain true. While we are investigating how to automatically generate 'adequate' lexicons (e.g. by adapting domain-independent ontologies, expanding domain-specific lists, or using techniques from automatic paraphrase generation), the question of how the lexicon is acquired and accessed (e.g. Querying over an API would require probing for named entities rather than simple hash look-up, etc.) is orthogonal to the contribution of this work.

We make **assumption c** because we want to guarantee that COVER lives up to the promise of always having within its non-empty result sets the correct interpretation. Still the control points for syntactic analysis are very clearly laid out in COVER. Given that interfaces should be usable by real users, keeping the interpretation set manageable while trying to keep the correct one in the set is useful, especially if a database is particularly ambiguous. Exploring methods to integrate off-the-shelf syntactic parsers to narrow the number of interpretations while not excluding correct interpretations will be future work. Specifically we will evaluate which off-the-shelf parsers and which notions of 'attachment' perform best.

A final question is, *is the semantically tractable class fundamental?* COVER has generalized the class from the earlier PRECISE work and nothing seems to block its further extension to questions requiring negation, circular self-joins, etc. Still, we expended considerable effort trying to extend the class to include queries with maximum cardinality conditions (e.g. "What are the states with the most cities?"). This effort foundered on defining a decidable semantics. But we also witnessed many cases where spurious valid mappings were let in while not finding a correct valid mapping ('most' seems to be a particularly sensitive word). Further study is required to determine the natural limit of the semantic tractability question class. How far can we go? Is there a limit? Our intuition says 'yes'. A related question is *is there a hierarchy of semantically tractable classes where the number of interpretations blows up as we extend the number of constructs we are able to handle?* Again, our intuition says 'yes'. COVER will be the basis of the future study of these questions.

References

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases. *NLE*, 1(1):29–81.

Ann A. Copestake and Karen Sparck Jones. 1990. Natural language interfaces to databases. *Knowledge Eng. Review*, 5(4):225–249.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *PVLDB*, 8(1):73–84.

Michael Minock. 2017. Technical description of COVER 1.0. *CoRR*.

Ana-Maria Popescu, Oren Etzioni, and Henry A. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *IUI-03, January 12-15, 2003, Miami, FL, USA*, pages 149 – 157.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases. In *COLING*, pages 141 – 147.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In *ISWC, Boston, MA, USA*, pages 362–374.

Guogen Zhang, Wesley W. Chu, Frank Meng, and Gladys Kong. 1999. Query formulation from high-level concepts for relational databases. In *UIDIS*, pages 64–75.

Common Round: Application of Language Technologies to Large-Scale Web Debates

Hans Uszkoreit, Aleksandra Gabryszak, Leonhard Hennig, Jörg Steffen, Renlong Ai, Stephan Busemann, Jon Dehdari, Josef van Genabith, Georg Heigold, Nils Rethmeier, Raphael Rubino, Sven Schmeier, Philippe Thomas, He Wang, Feiyu Xu
Deutsches Forschungszentrum für Künstliche Intelligenz, Germany
firstname.lastname@dfki.de

Abstract

Web debates play an important role in enabling broad participation of constituencies in social, political and economic decision-taking. However, it is challenging to organize, structure, and navigate a vast number of diverse argumentations and comments collected from many participants over a long time period. In this paper we demonstrate Common Round, a next generation platform for large-scale web debates, which provides functions for eliciting the semantic content and structures from the contributions of participants. In particular, Common Round applies language technologies for the extraction of semantic essence from textual input, aggregation of the formulated opinions and arguments. The platform also provides a cross-lingual access to debates using machine translation.

1 Introduction

Debating is a very useful approach to individual and collective decision making; it helps the formation of ideas and policies in democracies. Web-based debates allow for reaching a wider audience, therefore bringing more arguments and diverse perspectives on a debate topic compared to face-to-face discussions. The asynchronous mode of web debates also enables participants to explore debate content more thoroughly (Salter et al., 2016). However, these advantages often get lost in a large-scale debate since its content can become unmanageable. Moreover, missing background knowledge on debate topics or language barriers for international topics can prohibit users from a proper understanding of debate content.

The majority of the existing web discussion platforms offer the following participation model: users can formulate a discussion topic, share their opinions on that topic and respond to others in the form of unstructured posts. However, they do not offer effective functionalities for 1) easy access to the argumentative structure of debate content, and 2) quick overviews of the various semantic facets, the polarity and the relevance of the arguments. Some platforms[1] allow users to label posts as *pro* or *con* arguments, to cite external sources, to assess debate content or to create structured debates across the web, but do not offer any deeper automatic language technology-based analyses. Argumentation mining research, which could help in automatically structuring debates, has only recently been applied to web debate corpora (Boltuzic and Snajder, 2015; Petasis and Karkaletsis, 2016; Egan et al., 2016).

Our goal is to address these issues by developing a debate platform that:

- Supports debate participants in making substantial and clear contributions
- Facilitates an overview of debate contents
- Associates relevant information available on the web with debate topics
- Connects regional discussions to global deliberations
- Supports advanced participation in deliberations, without sacrificing transparency and usability.

In this paper, we present the Common Round platform, which implements various functions towards these goals, with following contributions: (a) we describe the architecture of the platform (Section 2), including a model for argumentative dialogues (Section 3) and (b) we present our implementation of several language technologies for supporting collective deliberation (Sections 4–6).

[1] Examples: ProCon.org, OpenPetition.de, Arvina and ArguBlogging (Bex et al., 2013)

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 5–8

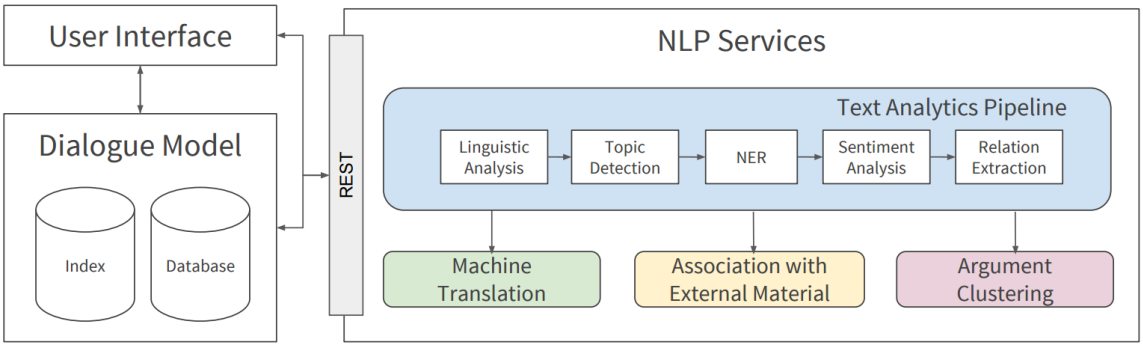


Figure 1: Architecture of Common Round.

2 Common Round platform - Overview

Figure 1 depicts the platform architecture. Common Round provides a top-level dialogue model that supports users to make decisions and enter labeled contributions at appropriate places in the model. This keeps debates organized and transparent, and at the same time allows users to participate in a free and unrestricted discussion. Furthermore a predefined coarse-grained debate structure facilitates the development of automatic text analysis algorithms for deliberation support, since it enables the system to integrate domain and context knowledge for better interpretation of user intentions. Within our platform, textual debate content is automatically further structured and enhanced employing argument mining, text analytics, retrieval of external material and machine translation. The results of these analyses are fed back to the front-end to give users a good overview of discussion content, and to enable a structured, semantic search for additional evidences and background material. We will exemplify the features of our platform with four selected topics *Should Cannabis be legalized?*, *Are German cars better than those from Japan?*, *Should we use wind power?*, *How to address online fake news?*

3 Dialogue model and content assessments

The Common Round platform provides a structure that aims to cover the most essential aspects of argumentative dialogues (Figure 2).

The top level defines the semantic categories of the debate questions. Given the categories, the questions themselves can be posted as the next level. There are two types of debate questions: (a) yes-no questions, and (b) multi-proposal questions (Table 1). A question reflects the major claim

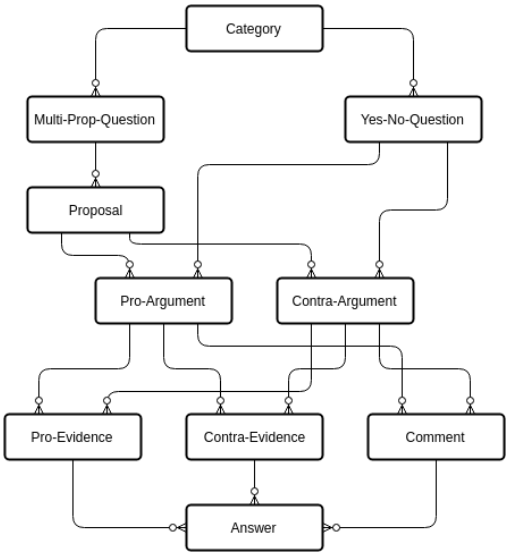


Figure 2: Dialogue model of Common Round.

for the debate. The subsequent level is reserved for

question type	example
Yes-No	Should Cannabis be legalized?
Multi-proposal	How to address online fake news?
Proposal 1	Impose sanctions on fake news authors.
Proposal 2	Impose sanctions on website providers.

Table 1: Yes-no and multi-proposal questions.

arguments. An argument can either be a *pro*- or a *con*-argument. For the yes-no questions, an argument directly refers to a *yes* answer to the question, while for multi-proposal questions an argument refers to a single proposal. At the next level, a user can add evidence in favor of or against an argument. Such evidence could be a textual contribution or a link to external sources. Finally, a user can freely answer to either an evidence or a comment or just refer to a previously posted answer. The length of the contributions is restricted in order to enforce separated, focused posts instead of

lengthy and noisy information.

Besides the dialogue structure, assessing postings is an important aspect in the Common Round forum. The assessment reflects the community view of a main debate claim, an argument or evidence. The degrees of consent a user can express are based on the common 5-level Likert-type scale (Likert, 1932). Based on the ratings a first time viewer of a question can get a quick overview of the state of the discussion.

4 Argument Mining

The dialogue model enables users to distinguish between argumentative and non-argumentative parts of a debate. Users can explore the components of argumentations, i.e. major claims, arguments and evidences. The categorization in *pro*/*con* arguments and *pro*/*con* evidences easily allows finding posts supporting or refuting the major claim. Furthermore, similar arguments are automatically clustered. To aggregate similar arguments we represent posts with Paragraph2Vec (Le and Mikolov, 2014) and then compute clusters from the cosine similarity between post vectors using agglomerative clustering. In order to keep the clustering unsupervised we automatically discover the number of clusters using the Silhouette Coefficient (Rousseeuw, 1987). To help users identify the most relevant content within a cluster, the arguments are sorted by their assessed validity. Reading only a few arguments from each cluster enables users to gain a quick overview of different argument facets. For example, in a debate on cannabis legalization, arguments from a medical point of view can be grouped separately from arguments that focus on economic reasons. This functionality is an important contribution to structuring large-scale debates.

5 Text Analytics and Association with External Material

The Common Round platform enriches the contents of debates and posts by extracting information about topics, sentiment, entities and relations. Topic detection helps to find semantically related debates. Sentiment analysis allows determining the emotion level in discussions. By identifying, for example, instances of the relation *MayTreatDisorder* in a discussion about the legalization of cannabis, we can aggregate references to the same or similar medical facts introduced by different debate participants. Additionally, our platform incorporates a web search to enable finding supporting evidence in external knowledge sources. Table 2 shows an example of the text analytics results and the association with external material.

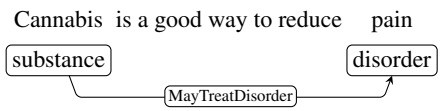
NER and RE	
Sentiment	positive
External Material	www.scientificamerican.com/article/could-medical-cannabis-break-the-painkiller-epidemic

Table 2: Example of the text analytics results.

Information extraction pipeline currently supports English and German texts. We utilize Stanford CoreNLP tools for segmentation, POS-tagging and dependency parsing of English texts. POS-tagging and dependency parsing of German texts are realized with Mate Tools. The topic detection is realized by the supervised document classification module PCL (Schmeier, 2013). PCL is proven to be very robust and accurate especially for short texts and unbalanced training data. For named entity recognition, we employ two complementary approaches. We apply Stanford NER models to recognize standard entity types, such as persons, organizations, locations and date/time expressions. For non-standard concept types, we use SProUT (Drozdzynski et al., 2004), which implements a regular expression-like rule formalism and gazetteers for detecting domain-specific concepts in text. Relation extraction is performed by matching dependency parse trees of sentences to a set of automatically learned dependency patterns (Krause et al., 2012). Relation patterns are learned from corpora manually annotated with event type, argument types, and roles, or using distant supervision (Mintz et al., 2009). For sentiment analysis, we apply a lexicon-based approach that additionally makes use of syntactic information in order to handle negation.

6 Cross-lingual Access

In order to support cross-lingual access to debate posts we developed a new character-based neural machine translation (NMT) engine and tuned on Common Round domains. Translation is available as a real-time service and translation outputs

are marked. Our NMT is based on an encoder–decoder with attention design (Bahdanau et al., 2014; Johnson et al., 2016), using bidirectional LSTM layers for encoding and unidirectional layers for decoding. We employ a character-based approach to better cope with rich morphology and OOVs in Common Round user posts. We train on English-German data from WMT-16[2] and use transfer learning on 30K crawled noisy in-domain segments for Common Round domain-adaptation. As our NMT system is new, we compare with s-o-t-a PBMT: while both perform in the top-ranks in WMT-16 (Bojar et al., 2016), NMT is better on Common Round data (NMT from 29.1 BLEU on WMT-16 data to 22.9 car and 23.8 wind power, against 30.0 to 13.6 and 17.7 for PBMT) and better adapts using the supplementary crawled data (25.8 car and 28.5 wind against 14.1 and 19.3).[3]

7 Conclusion & Future Work

We presented Common Round, a new type of web-based debate platform supporting large-scale decision making. The dialogue model of the platform supports users in making clear and substantial debate contributions by labeling their posts as *pro*/*con* arguments, *pro*/*con* evidences, comments and answers. The user input is automatically analyzed and enhanced using language technology such as argument mining, text analytics, retrieval of external material and machine translation. The analysis results are fed back to the user interface as an additional information layer. For future work we suggest a more fine-grained automatic text analysis, for example by detecting claims and premises of arguments as well as types and validity of evidences. Moreover we will conduct a thorough evaluation of the system.

Acknowledgments

This research was supported by the German Federal Ministry of Education and Research (BMBF) through the projects ALL SIDES (01IW14002) and BBDC (01IS14013E), by the German Federal Ministry of Economics and Energy (BMWi) through the projects SDW (01MD15010A) and SD4M (01MD15007B), and by the European Union's Horizon 2020 research and innovation programme through the project QT21 (645452).

[2] http://www.statmt.org/wmt16/translation-task.html

[3] All EN→DE, similar for reverse.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

F. Bex, J. Lawrence, M. Snaith, and C.Reed. 2013. Implementing the argument web. *Commun. ACM*, 56:66–73.

O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and Ma. Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proc. of MT*.

F. Boltuzic and J. Snajder. 2015. Identifying prominent arguments in online debates using semantic textual similarity. In *ArgMining@HLT-NAACL*.

W. Drozdzynski, H. Krieger, J. Piskorski, U. Schäfer, and F. Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

C. Egan, A. Siddharthan, and A. Z. Wyner. 2016. Summarising the points made in online political debates. In *ArgMining@ACL*.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. In *https://arxiv.org/abs/1611.04558*.

S. Krause, H. Li, H. Uszkoreit, and F. Xu. 2012. Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In *ISWC*.

Q. V. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

R. Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *IJCNLP*.

G. Petasis and V. Karkaletsis. 2016. Identifying argument components through TextRank. In *ArgMining@ACL*.

P. J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65.

S. Salter, T. Douglas, and D. Kember. 2016. Comparing face-to-face and asynchronous online communication as mechanisms for critical reflective dialogue. *JEAR*, 0(0):1–16.

S. Schmeier. 2013. *Exploratory Search on Mobile Devices*. DFKI-LT - Dissertation Series, Saarbruecken, Germany.

A Web-Based Interactive Tool for Creating, Inspecting, Editing, and Publishing Etymological Datasets

Johann-Mattis List
Max Planck Institute for the Science of Human History
Kahlaische Straße 10
07745 Jena
list@shh.mpg.de

Abstract

The paper presents the Etymological DICtionary ediTOR (EDICTOR), a free, interactive, web-based tool designed to aid historical linguists in creating, editing, analysing, and publishing etymological datasets. The EDICTOR offers interactive solutions for important tasks in historical linguistics, including facilitated input and segmentation of phonetic transcriptions, quantitative and qualitative analyses of phonetic and morphological data, enhanced interfaces for cognate class assignment and multiple word alignment, and automated evaluation of regular sound correspondences. As a web-based tool written in JavaScript, the EDICTOR can be used in standard web browsers across all major platforms.

1 Introduction

The amount of large digitally available datasets for various language families is constantly increasing. In order to analyse these data, linguists turn more and more to automatic approaches. Phylogenetic methods from biology are now regularly used to create evolutionary trees of language families (Gray and Atkinson, 2003). Methods for the comparison of biological sequences have been adapted and allow to automatically search for cognate words in multilingual word lists (List, 2014) and to automatically align them (List, 2014). Complex workflows are used to search for deep genealogical signals between established language families (Jäger, 2015).

In contrast to the large arsenal of software for automatic analyses, the number of tools helping to *manually* prepare, edit, and correct lexical datasets in historical linguistics is extremely rare. This is surprising, since automatic approaches still lag behind expert analyses (List et al., 2017). Tools for data preparation and evaluation would allow experts to directly interact with computational approaches by manually checking and correcting their automatically produced results. Furthermore, since the majority of phylogenetic approaches makes use of manually submitted expert judgments (Gray and Atkinson, 2003), it seems indispensable to have tools which ease these tasks.

2 The EDICTOR Tool

The Etymological DICtionary ediTOR (EDICTOR) is a free, interactive, web-based tool that was specifically designed to serve as an interface between quantitative and qualitative tasks in historical linguistics. Inspired by powerful features of STARLING (Starostin, 2000) and RefLex (Segerer and Flavier, 2015), expanded by innovative new features, and based on a very simple data model that allows for a direct integration with quantitative software packages like LingPy (List and Forkel, 2016), the EDICTOR is a lightweight but powerful toolkit for computer-assisted applications in historical linguistics.

2.1 File Formats and Data Structure

The EDICTOR was designed as a lightweight file-based tool that takes a text file as input, allowing to modify and save it. The input format is a plain tab-separated value (TSV) file, with a *header* indicating the value of the columns. This format is essentially identical with the format used in LingPy. Although the EDICTOR accepts all regular TSV files as input, its primary target are *multi-lingual word lists*, that is, datasets in which a given number of *concepts* has been translated into a certain range of *target languages*.

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 9–12

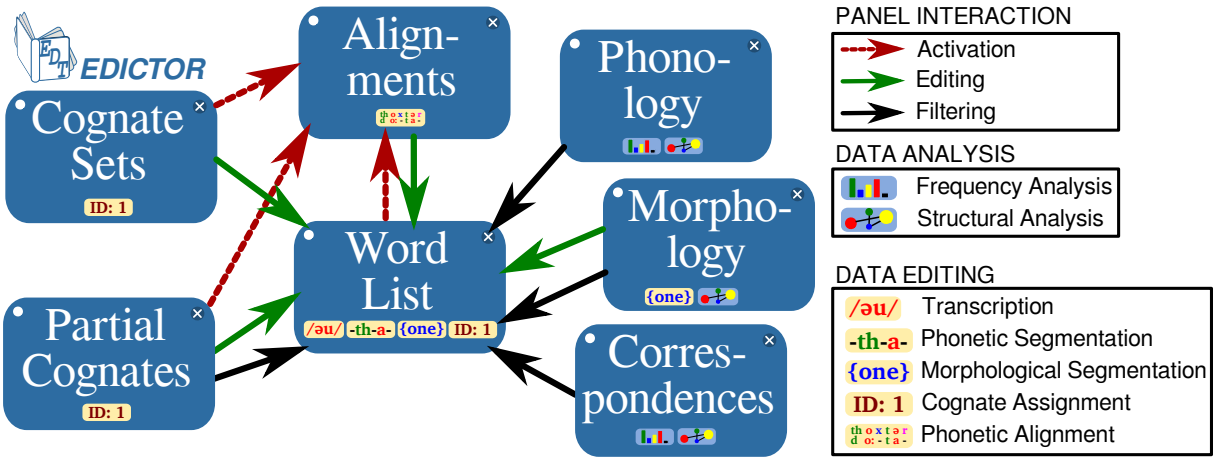


Figure 1: Basic panel structure of the EDICTOR.

ID	DOCULECT	CONCEPT	TRANSCRIPTION	...
1	German	Woldemort	valdəmar	...
2	English	Woldemort	wɔldəmɔrt	...
3	Chinese	Woldemort	fu^{51}ti^{51}mɔ^{35}	...
4	Russian	Woldemort	vladimir	...
...	...	...	...	...
10	German	Harry	haralt	...
11	English	Harry	hæri	...
12	Russian	Harry	gali	...
...	...	...	...	...

Figure 2: Basic file format in the EDICTOR

2.2 User Interface

The EDICTOR is divided into different *panels* which allow to edit or analyse the data in different ways. The core module is the *Word List panel* which displays the data in its original form and can be edited and analysed as one knows it from spreadsheet applications. For more complex tasks of data editing and analysis, such as cognate assignment or phonological analysis, additional panels are provided. Specific modes of interaction between the different panels allow for a flexible interaction between different tasks. Using drag-and-drop, users can arrange the panels individually or hide them completely. Figure 1 illustrates how the major panels of the EDICTOR interact with each other.

2.3 Technical Aspects

The EDICTOR application is written in plain JavaScript and was tested in Google Chrome, Firefox, and Safari across different operating systems (Windows, MacOS, Linux). For the purpose of offline usage, users can download the source code. For direct online usage, the tool can be accessed via its project website.

3 Data Editing in the EDICTOR

3.1 Editing Word List Data

Editing data in the Word List panel of the EDICTOR is straightforward by inserting values in text-fields which appear when clicking on a given field or when browsing the data using the arrow keys of the keyboard. Additional keyboard shortcuts allow for quick browsing. For specific data types, automatic operations are available which facilitate the input or test what the user inserts. Transcription, for example supports SAMPA-input. The segmentation of phonetic entries into meaningful sound units is also carried out automatically. Sound segments are highlighted with specific background colors based on their underlying sound class and sounds which are not recognized as valid IPA symbols are highlighted in warning colors (see the illustration in Figure 3). The users can decide themselves in which fields they wish to receive automatic support, and even Chinese input using an automatic Pīnyīn converter is provided.

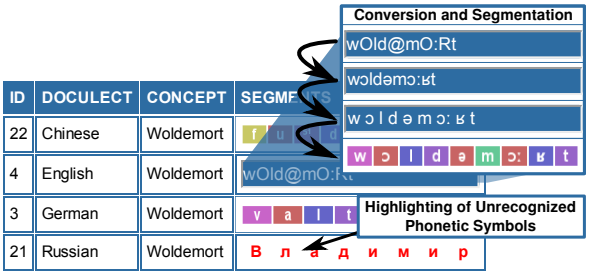


Figure 3: Editing word lists in the EDICTOR

3.2 Cognate Assessment

Defining which words in multilingual word lists are cognate is still a notoriously difficult task for machines (List, 2014). Given that the majority of datasets are based on manually edited cognate judgments, it is important to have tools which facilitate this task while at the same time controlling for typical errors. The EDICTOR offers two ways to edit cognate information, the first assuming complete cognacy of the words in their entirety, and the second allowing to assign only specific parts of words to the same cognate set. In order to carry out partial cognate assignment, the data needs to be morphologically segmented in a first stage, for example with help of the Morphology panel of the EDICTOR (see Section 4.2). For both tasks, simple and intuitive interfaces are offered which allow to browse through the data and to assign words to the same cognate set.

Figure 4: Aligning words in the EDICTOR

3.3 Phonetic Alignment

Since historical-comparative linguistics is essentially based on sequence comparison (List, 2014), alignment analyses, in which words are arranged in a matrix in such a way that corresponding sounds are placed in the same column, are underlying all cognate sets. Unfortunately they are rarely made explicit in classical etymological dictionaries. In order to increase explicitness, the EDICTOR offers an alignment panel. The alignment panel is essentially realized as a pop-up window showing the sounds of all sequences which belong to the same cognate set. Users can edit the alignments by moving sound segments with the mouse. Columns of the alignment which contain unalignable parts (like suffixes or prefixes) can be explicitly marked as such. In addition to manual alignments, the EDICTOR offers a simple alignment algorithm which can be used to pre-analyse the alignments. Figure 4 shows an example for the alignment of four fictive cognates in the EDICTOR.

4 Data Analysis in the EDICTOR

4.1 Analysing Phonetic Data

Errors are inevitable in large datasets, and this holds also and especially for phonetic transcriptions. Many errors, however, can be easily spotted by applying simple sanity checks to the data. A straightforward way to check the consistency of the phonetic transcriptions in a given dataset is provided in the Phonology panel of the EDICTOR. Here all sound segments which occur in the segmented transcriptions of one language are counted and automatically compared with an internal set of IPA segments. Counting the frequency of segments is very helpful to spot simple typing errors, since segments which occur only one time in the whole data are very likely to be errors. The internal segment inventory adds a structural perspective: If segments are found in the internal inventory, additional phonetic information (manner, place, etc.) is shown, if segments are missing, this is highlighted. The results can be viewed in tabular form and in form of a classical IPA chart.

4.2 Analysing Morphological Data

The majority of words in all languages consist of more than one morpheme. If historically related words differ regarding their morpheme structure, this poses great problems for automatic approaches to sequence comparison, since the algorithms usually compare words in their entirety. German *Großvater* 'grandfather', for example, is composed of two different morphemes, *groß* 'large' and *Vater* 'father'. In order to analyse multi-morphemic words historically, it is important to carry out a morphological annotation analysis. In order to ease this task, the Morphology panel of the EDICTOR offers a variety of straightforward operations by which morpheme structure can be annotated and analysed at the same time. The core idea behind all operations is a search for similar words or morphemes in the same language. These *colexifications* are then listed and displayed in form of a bipartite *word family network* in which words are linked to morphemes, as illustrated in Figure 5. The morphology analysis in the EDICTOR is no miracle cure for morpheme detection, and morpheme boundaries need still to be annotated by the user. However, the dynamically produced word family networks as well as the explicit listing of words sharing the same subsequence of sounds greatly facilitates this task.

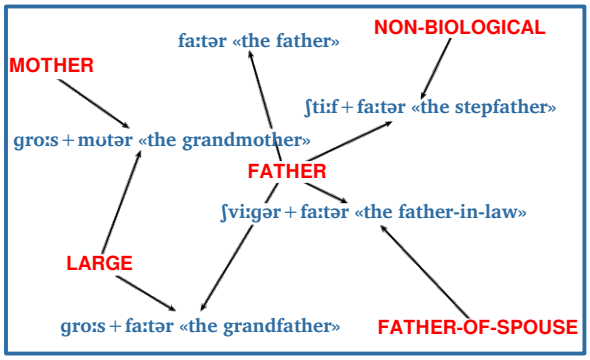


Figure 5: Word family network in the EDICTOR: The morphemes (in red) link the words around German *Großvater* 'grandfather' (in blue).

4.3 Analysing Sound Correspondences

Once cognate sets are identified and aligned, searching for regular sound correspondences in the data is a straightforward task. The Correspondences panel of the EDICTOR allows to analyse sound correspondence patterns across pairs of languages. In addition to a simple frequency count, however, *conditioning context* can be included in the analysis. Context is modeled as a separate string that provides abstract context symbols for each sound segment of a given word. This means essentially that context is handled as an additional *tier* of a sequence. This multi-tiered representation is very flexible and also allows to model suprasegmental context, like tone or stress. If users do not provide their own tiers, the EDICTOR employs a default context model which distinguishes consonants in syllable onsets from consonants in syllable offsets.

5 Customising the EDICTOR

The EDICTOR can be configured in multiple ways, be it while editing a dataset or before loading the data. The latter is handled via URL parameters passed to the URL that loads the application. In order to facilitate the customization procedure, a specific panel for customisation allows the users to define their default settings and creates a URL which users can bookmark to have quick access to their preferred settings.

The EDICTOR can be loaded in read-only mode by specifying a "publish" parameter. Additionally, server-side files can be directly loaded when loading the application. This makes it very simple and straightforward to use the EDICTOR to publish raw etymological datasets in a visually appealing format as can be seen from this exemplary URL: `http://edictor.digling.org?file=Tujia.tsv&publish=true&preview=500`.

6 Conclusion and Outlook

This paper presented a web-based tool for creating, inspecting, editing, and publishing etymological datasets. Although many aspects of the tool are still experimental, and many problems still need to be solved, I am confident that – even in its current form – the tool will be helpful for those working with etymological datasets. In the future, I will develop the tool further, both by adding more useful features and by increasing its consistency.

Acknowledgements

This research was supported by the DFG research fellowship grant 261553824 *Vertical and lateral aspects of Chinese dialect history*. I thank Nathan W. Hill, Guillaume Jacques, and Laurent Sagart for testing the prototype.

References

Russell D. Gray and Quentin D. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature*, 426(6965):435–439.

Gerhard Jäger. 2015. Support for linguistic macrofamilies from weighted alignment. *PNAS*, 112(41):1275212757.

Johann-Mattis List and Robert Forkel. 2016. *LingPy*. Max Planck Institute for the Science of Human History, Jena. URL: `http://lingpy.org`.

Johann-Mattis List, Simon J. Greenhill, and Russell D. Gray. 2017. The potential of automatic word comparison for historical linguistics. *PLOS ONE*, 12(1):1–18, 01.

Johann-Mattis List. 2014. *Sequence comparison in historical linguistics*. Düsseldorf University Press, Düsseldorf.

Guillaume Segerer and S. Flavier. 2015. *RefLex*. Laboratoire DDL, Paris and Lyon. URL: `http://reflex.cnrs.fr`.

Sergej Anatol'evič Starostin. 2000. *STARLING*. RGGU, Moscow. URL: `http://starling.rinet.ru`.

Supplementary Material

Supplements for this paper contain a demo video (`https://youtu.be/IyZuf6SmQM4`), the application website (`http://edictor.digling.org`), the source (v. 0.1, `https://zenodo.org/record/48834`), and the development version (`http://github.com/digling/edictor`).

WAT-SL: A Customizable Web Annotation Tool for Segment Labeling

Johannes Kiesel and **Henning Wachsmuth** and **Khalid Al-Khatib** and **Benno Stein**
Faculty of Media
Bauhaus-Universität Weimar
99423 Weimar, Germany
<first name>.<last name>@uni-weimar.de

Abstract

A frequent type of annotations in text corpora are labeled text segments. General-purpose annotation tools tend to be overly comprehensive, often making the annotation process slower and more error-prone. We present *WAT-SL*, a new web-based tool that is dedicated to segment labeling and highly customizable to the labeling task at hand. We outline its main features and exemplify how we used it for a crowdsourced corpus with labeled argument units.

1 Introduction

Human-annotated corpora are essential for the development and evaluation of natural language processing methods. However, creating such corpora is expensive and time-consuming. While remote annotation processes on the web such as crowdsourcing provide a way to obtain numerous annotations in short time, the bottleneck often lies in the used annotation tool and the required training of annotators. In particular, most popular annotation tools aim to be general purpose, such as the built-in editors of GATE (Cunningham, 2002) or web-based tools like BRAT (Stenetorp et al., 2012) and WebAnno (Yimam et al., 2014). Their comprehensiveness comes at the cost of higher interface complexity, which often also decreases the readability of the texts to be annotated.

Most of the typical annotation work is very focused, though, requiring only few annotation functionalities. An example is *segment labeling*, i.e., the assignment of one predefined label to each segment of a given text. Such labels may, e.g., capture clause-level argument units, sentence-level sentiment, or paragraph-level information extraction events. We argue that, for segment labeling, a dedicated tool is favorable in order to speed up the annotator training and the annotation process. While crowdsourcing platforms such as *mturk.com* or *crowdflower.com* follow a similar approach, their interfaces are either hardly customizable at all or need to be implemented from scratch.

This paper presents *WAT-SL* (Web Annotation Tool for Segment Labeling), an open-source web-based annotation tool dedicated to segment labeling.[1] WAT-SL provides all functionalities to efficiently run and manage segment labeling projects. Its self-descriptive annotation interface requires only a web browser, making it particularly convenient for remote annotation processes. The interface can be easily tailored to the requirements of the project using standard web technologies in order to focus on the specific segment labels at hand and to match the layout expectations of the annotators. At the same time, it ensures that the texts to be labeled remain readable during the whole annotation process. This process is server-based and preemptable at any point. The annotator's progress can be constantly monitored, as all relevant interactions of the annotators are logged in a simple key-value based plain text format.

In Section 2, we detail the main functionalities of WAT-SL, and we explain its general usage. Section 3 then outlines how we customized and used WAT-SL ourselves in previous work to label over 35,000 argumentative segments in a corpus with 300 news editorials (Al-Khatib et al., 2016).

2 Segment Labeling with WAT-SL

WAT-SL is a ready-to-use and easily customizable web-based annotation tool that is dedicated to segment labeling and that puts the focus on easy usage for all involved parties: annotators, annotation curators, and annotation project organizers.

[1] WAT-SL is available open source under a MIT license at: https://github.com/webis-de/wat

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 13–16

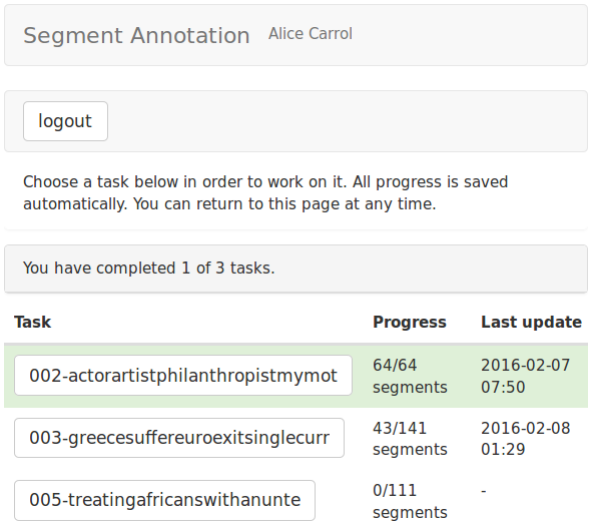


Figure 1: Screenshot of an exemplary task selection page in WAT-SL, listing three assigned tasks.

In WAT-SL, the annotation process is split into *tasks*, usually corresponding to single texts. Together, these tasks form a *project*.

2.1 Annotating Segments

The annotation interface is designed with a focus on easy and efficient usage. It can be accessed with any modern web browser. In order to start, annotators require only a login and a password.

When logging in, an annotator sees the *task selection page* that lists all assigned tasks, including the annotator's current progress in terms of the number of labeled segments (Figure 1). Completed tasks are marked green, and the web page automatically scrolls down to the first uncompleted task. This allows annotators to seamless interrupt and continue the annotation process.

After selecting a task to work on, the annotator sees the main *annotation interface* (Figure 2). The design of the interface seeks for clarity and self-descriptiveness, following the templates of today's most popular framework for responsive web sites, *Bootstrap*. As a result, we expect that many annotators will feel familiar with the style.

The central panel of the annotation interface shows the text to be labeled and its title. One design objective was to obtain a non-intrusive annotation interface that remains close to just displaying the text in order to maximize readability. As shown in Figure 2, we decided to indicate segments only by a shaded background and a small button at the end. To ensure a natural text flow, line breaks are possible within segments. The button reveals a menu for selecting the label. When moving the mouse cursor over a label, the label description is displayed to prevent a faulty selection. Once a segment is labeled, its background color changes, and the button displays an abbreviation of the respective label. To assist the annotators in forming a mental model of the annotation interface, the background colors of labeled segments match the label colors in the menu. All labels are saved automatically, avoiding any data loss in case of power outages, connection issues, or similar.

In some cases, texts might be over-segmented, for example due to an automatic segmentation. If this is the case, WAT-SL allows annotators to mark a segment as being continued in the next segment. The interface will then visually connect these segments (cf. the buttons showing "->" in Figure 2).

Finally, the annotation interface includes a text box for leaving comments to the project organizers. To simplify the formulation of comments, each segment is numbered, with the number being shown when the mouse cursor is moved over it.

2.2 Curating Annotations

After an annotation process is completed, a curation phase usually follows where the annotations of different annotators are consolidated into one. The WAT-SL *curation interface* enables an efficient curation by mimicking the annotation interface with three adjustments (Figure 3): First, segments for which the majority of annotators agreed on a label are pre-labeled accordingly. Second, the menu shows for each label how many annotators chose it. And third, the label description shows (anonymized) which annotator chose the label, so that curators can interpret each label in its context.

The curation may be accessed under the same URL as the annotation in order to allow annotators of some tasks being curators of other tasks.

2.3 Running an Annotation Project

WAT-SL is a platform-independent and easily deployable standalone Java application, with few configurations stored in a simple "key = value" file. Among others, annotators are managed in this file by assigning a login, a password, and a set of tasks to each of them. For each task, the organizer of an annotation project creates a directory (see below). WAT-SL uses the directory name as the task name in all occasions. Once the Java archive file we provide is then executed, it reads all configurations and starts a server. The server is immediately ready to accept requests from the annotators.

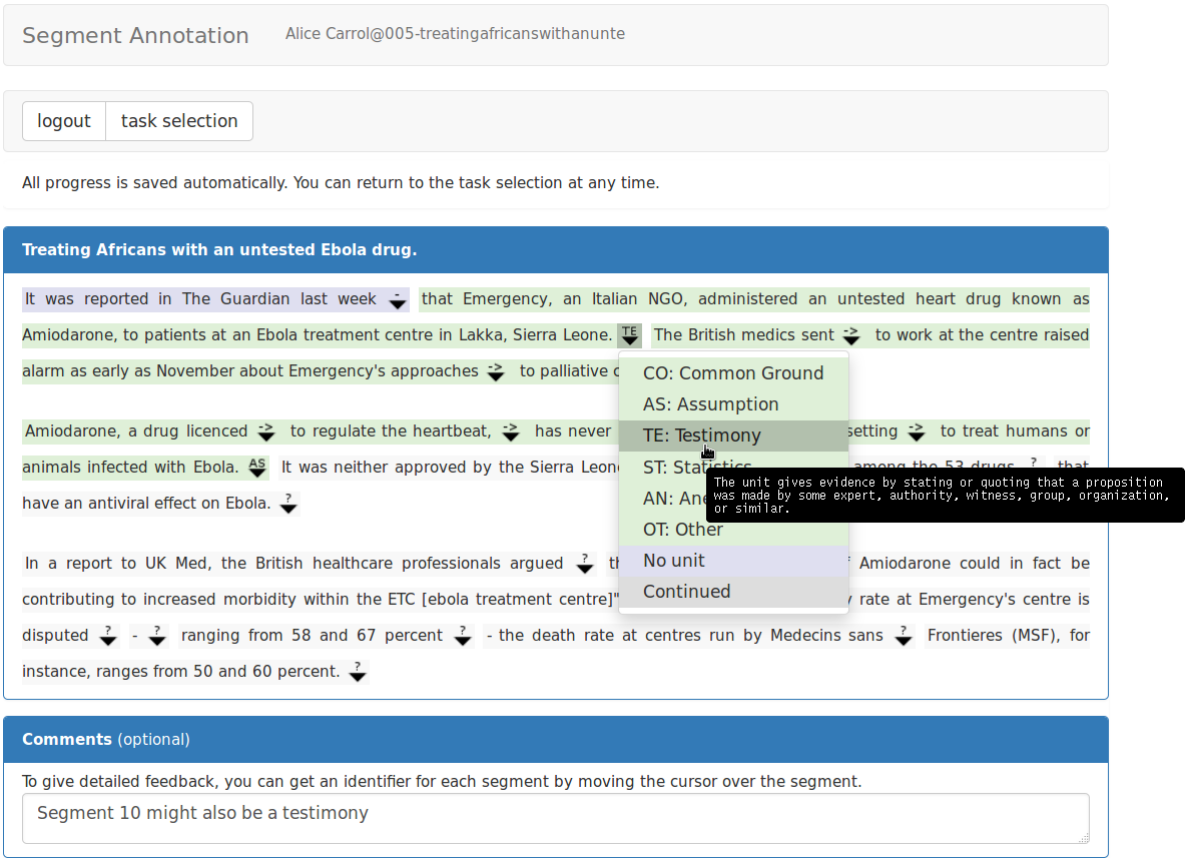


Figure 2: Screenshot of the annotation interface of WAT-SL, capturing the annotation of one news editorial within the argumentative segment labeling project described in Section 3. In particular, the screenshot illustrates how the annotator selects a label (*Testimony*) for one segment of the news editorial.

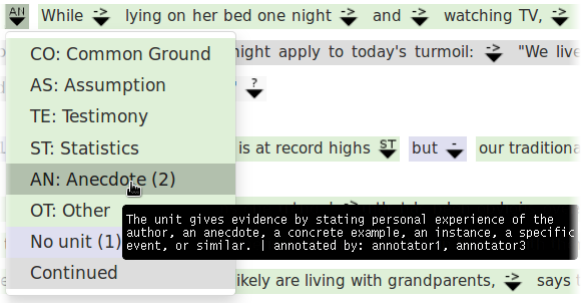


Figure 3: Screenshot of the curation interface, illustrating how a label (*Anecdote*) is selected based on counts of all labels the annotators selected for the respective segment (*Anecdote (2)*, *No unit (1)*).

WAT-SL logs all relevant annotator interactions. Whenever an annotator changes a segment label, the new label is immediately sent to the server. This prevents data loss and allows to monitor the progress. In addition to the new label, WAT-SL logs the current date, and the time offset and IP address of the annotator. With these logs, researchers can analyze the annotators' behavior, e.g., to identify hard cases where annotators needed much time or changed the label multiple times.

Annotating Segments Only few steps are needed to set up the segment labeling tasks: (1) List the labels and their descriptions in the configuration file, and place the button images in the corresponding directory. (2) Set the title displayed above each text (see Figure 2) in a separate configuration file in the respective task directory, or project-wide in the same file as the labels. (3) Finally, put the texts in the task directories. For easy usage, the required text format is as simple as possible: one segment per line and empty lines for paragraph breaks. Optionally, organizers can add Cascading Style Sheet and JavaScript files to customize the interface.

Curating Annotations To curate a task, an organizer duplicates the annotation task and then copies the annotation logs into the new task directory. The organizer then specifies curators for the curation task analog to assigning annotators.

Result In addition to the logs, the web interface also allows the organizer to see the final labels without history in a simple "key = value" format, which is useful when distributing the annotations.

3 Case Study: Labeling Argument Units

WAT-SL was already used successfully in the past, namely, for creating the *Webis-Editorials-16 corpus* with 300 news editorials split into a total of 35,665 segments, each labeled by three annotators and finally curated by the authors (Al-Khatib et al., 2016). In particular, each segment was assigned one of eight labels, where the labels include six types of argument units (e.g., *assumption* and *anecdote*), a label for non-argumentative segments, and a label indicating that a unit is continued in the next segment (see the "->" label in Section 2). On average, each editorial contains 957 tokens in 118 segments.

The annotation project of the Webis-Editorials-16 corpus included one task per editorial. The editorial's text had been pre-segmented with a heuristic unit segmentation algorithm before. This algorithm was tuned towards oversegmenting a text in case of doubt, i.e., to avoid false negatives (segments that should have been split further) at the cost of more false positives (segments that need to be merged). Note that WAT-SL allows fixing such false positives using "->".

For annotation, we hired four workers from the crowdsourcing platform *upwork.com*. Given the segmented editorials, each worker iteratively chose one assigned editorial (see Figure 1), read it completely, and then selected the appropriate label for each segment in the editorial (see Figure 2). This annotation process was repeated for all editorials, with some annotators interrupting their work on an editorial and returning to it later on. All editorials were labeled by three workers, resulting in 106,995 annotations in total. The average time per editorial taken by a worker was ~20 minutes.

To create the final version of the corpus, we curated each editorial using WAT-SL. In particular, we automatically kept all labels with majority agreement, and let one expert decide on the others. Also, difficult cases where annotators tend to disagree were identified in the curation phase.

From the perspective of WAT-SL, the annotation of the Webis-Editorials-16 corpus served as a case study, which provided evidence that the tool is easy to learn and master. From the beginning, the workers used WAT-SL without noteworthy problems, as far as we could see from monitoring the interaction logs. Also, the comment area turned out to be useful, i.e., the workers left several valuable suggestions and questions there.

4 Conclusion and Future Work

This paper has presented WAT-SL, an open-source web annotation tool dedicated to segment labeling. WAT-SL is designed for easily configuration and deployment, while allowing project organizers to tailor its annotation interface to the project needs using standard web technologies. WAT-SL aims for simplicity by featuring a self-explanatory interface that does not distract from the text to be annotated, as well as by always showing the annotation progress at a glance and saving it automatically. The curation of the annotations can be done using exactly the same interface. In case of oversegmented texts, annotators can merge segments. Moreover, all relevant annotator interactions are logged, allowing projects organizers to monitor and analyze the annotation process.

Recently, WAT-SL was used successfully on a corpus with 300 news editorials, where four remote annotators labeled 35,000 argumentative segments. In the future, we plan to create more dedicated annotation tools in the spirit of WAT-SL for other annotation types (e.g., relation identification). In particular, we plan to improve the annotator management functionalities of WAT-SL and reuse them for these other tools.

References

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A News Editorial Corpus for Mining Argumentation Strategies. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 3433–3443.

Hamish Cunningham. 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36(2):223–254.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Seid Muhie Yimam, Chris Biemann, Richard Eckart de Castilho, and Iryna Gurevych. 2014. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96.

TextImager as a Generic Interface to R

Tolga Uslu[1], Wahed Hemati[1], Alexander Mehler[1], and Daniel Baumartz[2]

[1,2]Goethe University of Frankfurt
[1,2]TextTechnology Lab
[1]{uslu,hemati,mehler}@em.uni-frankfurt.de
[2]baumartz@stud.uni-frankfurt.de

Abstract

R is a very powerful framework for statistical modeling. Thus, it is of high importance to integrate R with state-of-the-art tools in NLP. In this paper, we present the functionality and architecture of such an integration by means of *TextImager*. We use the *OpenCPU API* to integrate *R* based on our own *R*-Server. This allows for communicating with *R*-packages and combining them with *TextImager*'s NLP-components.

1 Introduction

We introduced *TextImager* in (Hemati et al., 2016) where we focused on its architecture and the functions of its backend. In this paper, we present the functionality and architecture of *R* interfaced by means of *TextImager*. For this purpose, we created a separate panel in *TextImager* for *R*-applications. In this panel, we combine state-of-the-art NLP tools embedded into *TextImager* with the the powerful statistics of *R* (R Development Core Team, 2008). We use the *OpenCPU API* (Ooms, 2014) to integrate *R* into *TextImager* by means of our own *R*-server. This allows for easily communicating with the built-in *R*-packages and combining the advantages of both worlds. In the case of topic detection (`LDA`), for example, the complete text is used as an input string to *R*. Thanks to *TextImager*'s preprocessor, more information is provided about syntactic words, parts of speech, lemmas, grammatical categories etc., which can improve topic detection. Further, the output of R routines is displayed by means of *TextImager*'s visualizations, all of which are linked to the corresponding input text(s). This allows for unprecedented *interaction* between text and the statistical results computed by *R*. For this paper we sampled several Wikipedia articles to present all features of *R* integrated into *TextImager*. This includes articles about four politicians (*Angela Merkel*, *Barack Obama*, *Recep (Tayyip) Erdoğan*, *Donald Trump*) and five sportsman, that is, three basketball players (*Michael Jordan*, *Kobe Bryant* and *Lebron James*) and two soccer players (*Thomas Müller* and *Bastian Schweinsteiger*).

2 Related Work

R is used and developed by a large community covering a wide range of statistical packages. The *CRAN*[1] package repository is the main repository of the *R* project. It currently consists of about 10 000 packages including packages for NLP.[2] The *R* framework requires basic to versatile skills in programming and scripting. It provides limited visualization and interaction functionalities. Attempts have been undertaken to provide web interfaces for *R* as, for example, *Shiny*[3] and *rApache*[4]. Though they provide a variety of functions and visualizations[5], these tools are not optimized for statistical NLP: their NLP-related functionalities are rather limited. In order to fill this gap, we introduce *TextImager*'s *R* package, that is, a web based tool for NLP utilizing *R*.

3 Architecture

TextImager is a *UIMA*-based (Ferrucci and Lally, 2004) framework that offers a wide range of NLP and visualization tools by means of a user-friendly *GUI* without requiring programming skills. It consists of two parts: front-end and back-end. The back-end is a modular, expandable, scalable and

[1] `https://cran.r-project.org/`
[2] `https://cran.r-project.org/web/views/NaturalLanguageProcessing.html`
[3] `http://shiny.rstudio.com/`
[4] `http://rapache.net/`
[5] `http://shiny.rstudio.com/gallery/`

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 17–20

flexible architecture with parallel and distributed processing capabilities (Hemati et al., 2016). The front-end is a web application that makes NLP processes available in a user-friendly way with responsive and interactive visualizations (Hemati et al., 2016). *TextImager* already integrated many third party tools. One of these is *R*. This section describes the technical integration and utilization of *R* into *TextImager*.

3.1 R / OpenCPU

R is a software environment for statistical computing. It compiles and runs on a variety of UNIX and Windows platforms. One of our goals is to provide an easy to use interface for R with a focus on NLP. To this end, we use *OpenCPU* to integrate *R* into *TextImager*. *OpenCPU* provides an *HTTP API*, which allocates the functionalities of R-packages (Ooms, 2014). The *OpenCPU* software can be used directly in *R*; alternatively, it can be installed on a server. We decided for the latter variant. In addition, we used the *opencpu.js* JavaScript library which simplifies API use in JavaScript and allows for calling *R*-functions directly from *TextImager*. To minimize the communication effort between client and server and to encapsulate *R* scripts from *TextImager*'s functionality, we created a so called *TextImager-R*-package that takes *TextImager* data, performs all *R*-based calculations and returns the results. This package serves for converting any *TextImager* data to meet the input requirements of any *R*-package. In this way, we can easily add new packages to *TextImager* without changing the HTTP request code. Because some data and models have a long build time we used *OpenCPU*'s session feature to keep this data on the server and access it in future sessions so that we do not have to recreate it. This allows the user for quickly executing several methods even in parallel without recalculating them each time.

3.2 Data Structure

The data structure of *TextImager* differs from *R*'s data structure. Therefore, we developed a generic mapping interface that translates the data structure from *TextImager* to an *R*-readable format. Depending on the *R*-package, we send the required data via *OpenCPU*. This allows for combining each NLP tool with any *R*-package.

3.3 OpenCPU Output Integration

Visualizing the results of a calculation or sensitivity analysis is an important task. That is why we provide interactive visualizations to make the information available to the user more comprehensible. This allows the user to interact with the text and the output of R, for example, by highlighting any focal sentence in a document by hovering over a word or sentence graph generated by means of R.

3.4 R-packages

This section gives an overview of *R*-packages embedded into the pipeline of *TextImager*.

tm The *tm*-package (Feinerer and Hornik, 2015) is a text mining *R*-package containing preprocess methods for data importing, corpus handling, stopword filtering and more.

lda The *lda*-package (Chang, 2015) provides an implementation of *Latent Dirichlet Allocation* algorithms. It is used to automatically identify topics in documents, to get top documents for these topics and to predict document labels. In addition to the tabular output we developed an interactive visualization, which assigns the decisive words to the appropriate topic (see Figure 1). This visualization makes it easy to differentiate between the two topics and see which words classify these topics. In our example, we have recognized words such as *players*, *season* and *game* as one topic (sportsman) and *party*, *state* and *politically* as a different topic (politics). The parameters of every package can be set on runtime. The utilization and combination of *TextImager* and *R* makes it possible, to calculate topics not only based on wordforms, but also takes other features into account like lemma, pos-tags, morphological features and more.

stylo The *stylo*-package (Eder et al., 2016) provides functionality for stylometric analyses. All parameters of the package can be set through the graphical user interface of TextImager. The package provides multiple unsupervised analyses, mostly based on a most-frequent-word list and contrastive text analysis. In Figure 2 we have calculated a cluster analysis based on our example corpus. We can see that the politicians, basketball players

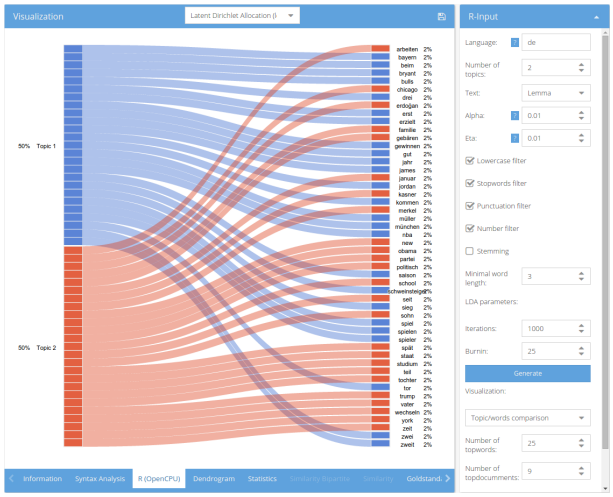


Figure 1: Words assigned to their detected topics

and soccer players have been clustered into their own groups.

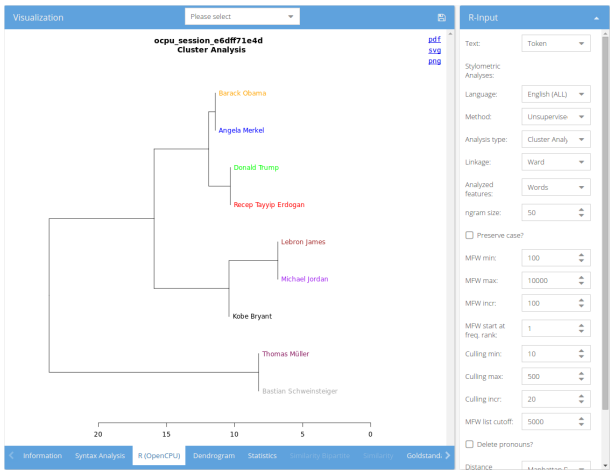


Figure 2: Cluster analysis of the documents

LSAfun The *LSAfun* (Günther et al., 2015) and *lsa* (Wild, 2015) packages provide functionality for *Latent Semantic Analysis*. In *TextImager* it is used to generate summaries of documents and similarities of sentences.

igraph The *igraph*-package (Csardi and Nepusz, 2006) provides multiple network analysis tools. We created a document network by linking each word with its five most similar words based on word embeddings. The *igraph*-package also allows to layout the graph and calculate different centrality measures. In the end, we can export the network in many formats (*GraphML*, *GEXF*, *JSON*, etc.) and edit it with graph editors. We also build an interactive network visualization (see figure 3) using *sigma.js* to make it interoperable with *TextImager*.

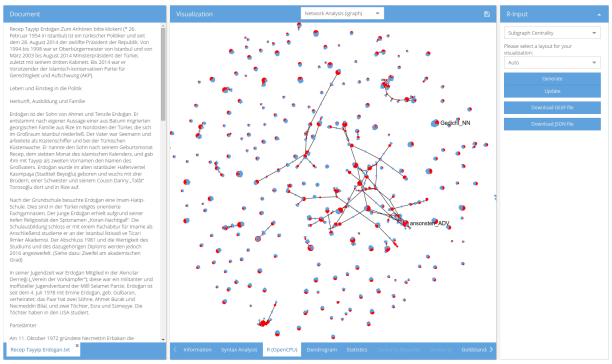
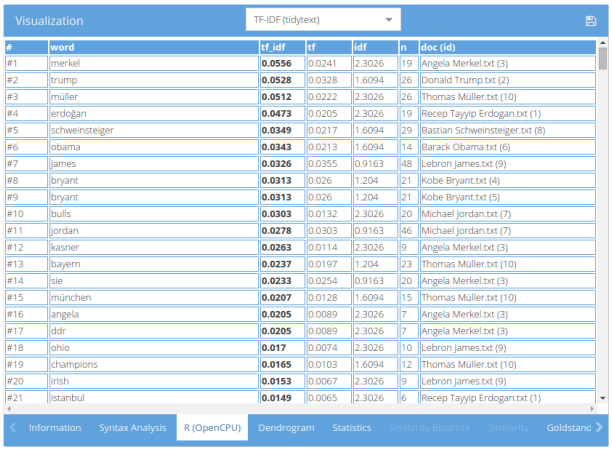


Figure 3: Network graph based on word embeddings

tidytext The *tidytext* package (Silge and Robinson, 2016) provides functionality to create datasets following the tidy data principles (Wickham, 2014). We used it with our *tm* based corpus to calculate *TF-IDF* information of documents. In Figure 4 we see the output tabular with informations like *tf*, *idf* and *tf-idf*



#	word	tf_idf	tf	idf	n	doc (id)
#1	merkel	**0.0556**	0.0241	2.3026	19	Angela Merkel.txt (3)
#2	trump	**0.0528**	0.0328	1.6094	26	Donald Trump.txt (2)
#3	müller	**0.0512**	0.0222	2.3026	26	Thomas Müller.txt (10)
#4	erdoğan	**0.0473**	0.0205	2.3026	19	Recep Tayyip Erdogan.txt (1)
#5	schweinsteiger	**0.0349**	0.0217	1.6094	29	Bastian Schweinsteiger.txt (8)
#6	obama	**0.0343**	0.0213	1.6094	14	Barack Obama.txt (6)
#7	james	**0.0326**	0.0355	0.9163	48	Lebron James.txt (9)
#8	bryant	**0.0313**	0.026	1.204	21	Kobe Bryant.txt (4)
#9	bryant	**0.0313**	0.026	1.204	21	Kobe Bryant.txt (5)
#10	bulls	**0.0303**	0.0132	2.3026	20	Michael Jordan.txt (7)
#11	jordan	**0.0278**	0.0303	0.9163	46	Michael Jordan.txt (7)
#12	kasner	**0.0263**	0.0114	2.3026	9	Angela Merkel.txt (3)
#13	bayern	**0.0237**	0.0197	1.204	23	Thomas Müller.txt (10)
#14	sie	**0.0233**	0.0254	0.9163	20	Angela Merkel.txt (3)
#15	münchen	**0.0207**	0.0128	1.6094	15	Thomas Müller.txt (10)
#16	angela	**0.0205**	0.0089	2.3026	7	Angela Merkel.txt (3)
#17	ddr	**0.0205**	0.0089	2.3026	7	Angela Merkel.txt (3)
#18	ohio	**0.017**	0.0074	2.3026	10	Lebron James.txt (9)
#19	champions	**0.0165**	0.0103	1.6094	12	Thomas Müller.txt (10)
#20	irish	**0.0153**	0.0067	2.3026	9	Lebron James.txt (9)
#21	istanbul	**0.0149**	0.0065	2.3026	6	Recep Tayyip Erdogan.txt (1)

Figure 4: Statistical information of the documents

stringdist The *stringdist*-package (van der Loo, 2014) implements distance calculation methods, like `Cosine`, `Jaccard`, `OSA` and other. We implemented functionality for calculating sentence similarities and provide an interactive visual representation. Each node represents an sentence of the selected document and the links between them represent the similarity of those sentences. The thicker the links, the more similar they are. By in-

teracting with the visualization, corresponding sections of the document panel are getting highlighted, to see the similar sentences (see Figure 5). The bidirectional interaction functionality enables easy comparability.

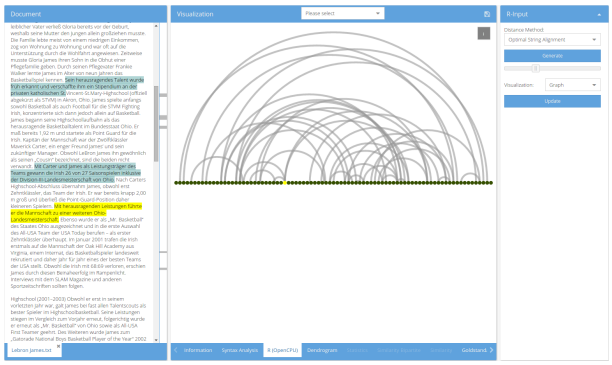


Figure 5: Depiction of sentence similarities.

stats We used functions from the *R*-package *stats* (R Development Core Team, 2008) to calculate a hierarchical cluster analysis based on the sentence similarities. This allows us to cluster similar sentences and visualize them with an interactive dendrogram. In Figure 6 we selected on one of these clusters and the document panel immediately adapts and highlights all the sentences in this cluster.

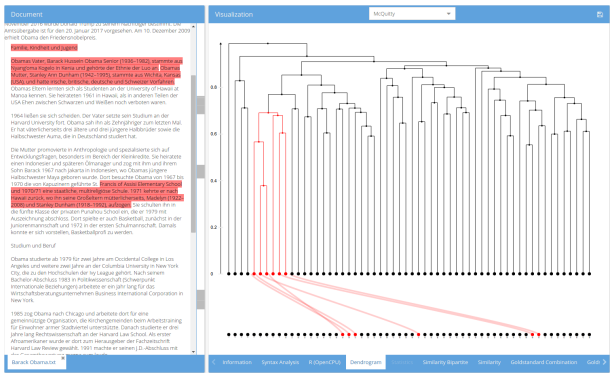


Figure 6: Similarity-clustered sentences.

An interesting side effect of integrating these tools into TextImager's pipeline is that their output can be concerted in a way to arrive at higher-level text annotations and analyses. In this way, we provide to an integration of two heavily expanding areas, that is, NLP and statistical modeling.

4 Future work

In already ongoing work, we focus on big data as exemplified by the Wikipedia. We also extend the number of built-in *R*-packages in *TextImager*.

5 Scope of the Software Demonstration

The beta version of *TextImager* is online. To test the functionalities of *R* as integrated into *TextImager* use the following demo: `http://textimager.hucompute.org/index.html?viewport=demo&file=R-Demo.xml`.

References

Jonathan Chang, 2015. *lda: Collapsed Gibbs Sampling Methods for Topic Models*. R package version 1.4.2.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Maciej Eder, Jan Rybicki, and Mike Kestemont. 2016. Stylometry with R: a package for computational text analysis. *R Journal*, 8(1):107–121.

Ingo Feinerer and Kurt Hornik, 2015. *tm: Text Mining Package*. R package version 0.6-2.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Fritz Günther, Carolin Dudschig, and Barbara Kaup. 2015. Lsafun: An R package for computations based on latent semantic analysis. *Behavior Research Methods*, 47(4):930–944.

Wahed Hemati, Tolga Uslu, and Alexander Mehler. 2016. Textimager: a distributed UIMA-based system for NLP. In *Proceedings of the COLING 2016 System Demonstrations*.

Jeroen Ooms. 2014. The OpenCPU system: Towards a universal interface for scientific computing through separation of concerns. *arXiv:1406.4806 [stat.CO]*.

R Development Core Team, 2008. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Julia Silge and David Robinson. 2016. tidytext: Text mining and analysis using tidy data principles in R. *JOSS*, 1(3).

M.P.J. van der Loo. 2014. The stringdist package for approximate string matching. *The R Journal*, 6:111–122.

Hadley Wickham. 2014. Tidy data. *Journal of Statistical Software*, 59(1):1–23.

Fridolin Wild, 2015. *lsa: Latent Semantic Analysis*. R package version 0.73.1.

GraWiTas: a Grammar-based Wikipedia Talk Page Parser

Benjamin Cabrera
University of Duisburg-Essen
Lotharstr. 65
47057 Duisburg, Germany

Laura Steinert
University of Duisburg-Essen
Lotharstr. 65
47057 Duisburg, Germany

Björn Ross
University of Duisburg-Essen
Forsthausweg 2
47057 Duisburg, Germany

firstname.lastname@uni-due.de

Abstract

Wikipedia offers researchers unique insights into the collaboration and communication patterns of a large self-regulating community of editors. The main medium of direct communication between editors of an article is the article's talk page. However, a talk page file is unstructured and therefore difficult to analyse automatically. A few parsers exist that enable its transformation into a structured data format. However, they are rarely open source, support only a limited subset of the talk page syntax – resulting in the loss of content – and usually support only one export format. Together with this article we offer a very fast, lightweight, open source parser with support for various output formats. In a preliminary evaluation it achieved a high accuracy. The parser uses a *grammar*-based approach – offering a transparent implementation and easy extensibility.

1 Introduction

Wikipedia is becoming an increasingly important knowledge platform. As the content is created by a self-regulating community of users, the analysis of interactions among users with methods from natural language processing and social network analysis can yield important insights into collaboration patterns inherent to such platforms. For example, Viegas et al. (2007) manually classified talk page posts with regard to the communication type to analyse the coordination among editors. Such insights can be important for research in the area of Computer-Supported Collaborative Learning.

The collaboration patterns among Wikipedia editors are visible on an article's *revision history* and its *talk page*. The revision history lists all changes ever made to an article, but it does not contain explicit information about the collaboration between editors. Most discussions between editors take place on the article's talk page, a dedicated file where they can leave comments and discuss potential improvements and revisions.

The talk page is therefore useful for observing explicit coordination between editors. However, most studies on Wikipedia article quality have focused on easily accessible data (Liu and Ram, 2011), whereas talk pages are not easy to use with automated methods. Essentially, talk pages are very loosely structured text files for which the community has defined certain formatting rules. Editors do not always follow these rules in detail and thus an automated analysis of talk page data requires a parsing of the file into a structured format by breaking it down into the individual comments and the links among them.

In this article, we introduce an open source parser for article talk pages called *GraWiTas* that focuses on a good comment detection rate, good usability and a plethora of different output formats. While a few of such talk page parsers exist, our core parser is based on the Boost.Spirit C++-library[1] which utilises Parsing Expression Grammars (PEG) for specifying the language to parse. This leads to a very fast, easily extensible program for different use cases.

The next section of this paper describes the structure of Wikipedia talk pages. The third section describes the parser we developed. Afterwards, a preliminary evaluation is described. The fifth section gives an overview on related work. Finally, conclusions from our research are given at the end of the paper.

[1] http://boost-spirit.com/, as seen on Feb. 14, 2017

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 21–24

2 Wikipedia Talk Pages

The talk page of a Wikipedia article is the place where editors can discuss issues with the article content and plan future contributions. There is a talk page for every article on Wikipedia. As any other page in Wikipedia, it can be edited by anybody by manipulating the underlying text file – which uses *Wiki markup*, a dedicated markup syntax. When a user visits the talk page, this syntax file is interpreted by the Wikipedia template engine and turned into the displayed HTML.

Although Wiki markup includes many different formatting (meta-)commands, the commands used on talk pages are fairly basic. Some guidelines as to how to comment on talk pages are defined in the *Wikipedia Talk Page Guidelines*[2]. These rules specify, for example, that new topics should be added to the bottom of the page, that a user may not alter or delete another user's post, that a user should sign a post with their username and a timestamp, and that to indicate to which older post they are referring, users should indent their own post.

The following snippet gives an example of the talk page syntax:

```
== Meaning of second paragraph ==
I don't understand the second paragraph
    ... [[User:U1|U1]] [[User Talk:U1|
    talk]] 07:24, 2 Dec 2016 (UTC)
:I also find it confusing...[[User:U2|U2
    ]] 17:03, 3 Dec 2016 (UTC)
::Me too... [[User:U3|U3]] [[User Talk:
    U3|talk]] 19:58, 3 Dec 2016 (UTC)
:LOL, the unit makes no sense [[User:U4|
    U4]] [[User Talk:U1|talk]] 00:27, 6
    Dec 2016 (UTC)
Is the reference to source 2 correct? [[
    User:U3|U3]] [[User Talk:U3|talk]]
    11:41, 4 Dec 2016 (UTC)
```

The first line marks the beginning of a new discussion topic on the page. The following lines contain comments on this topic. All authors signed their comments with a signature consisting of some variation of a link to their user profile page – in Wiki markup, hyperlinks are wrapped in '[[' and ']]' – and a timestamp. User U2 replies to the first post and thus indents their text by one tab. In Wiki markup this is done with a colon ':'. The third comment, which is a reply to the second one, is indented by two colons, leading to more indentation on the page when it is displayed.

While this structure is easily understood by humans, parsing talk page discussions automatically is far from trivial for a number of reasons. The discussion is not stored in a structured database format, and many editors use slight variations of the agreed-upon syntax when writing their comments. For example, there are multiple types of signatures that mark the end of a comment. Some editors do not indent correctly or use other formatting commands in the Wikipedia arsenal.

In addition, some Wikipedia talk pages contain *transcluded* content. This means that the content of a different file is pasted into the file at the specified position. Also, pages that become "too large" are *archived*[3], meaning that the original page is moved to a different name and a new, empty page is created in its place.

3 GraWiTas

Our parser – GraWiTas – consists of three components, covering the whole process of retrieving and parsing Wikipedia talk pages. The first two components gather and preprocess the needed data. They differ in the used data source: The *crawler component* extracts the talk page content of given Wikipedia URLs while the *dump component* processes full Wikipedia XML dumps. The actual parsing is done in the *core parser component*, where all comments from the Wiki markup of a talk page are extracted and exported into one of several output formats.

3.1 Core Parser Component

The main logic of the core parser lies in a system of rules that essentially defines what to consider as a comment on a talk page. Simply spoken, a comment is a piece of text – possibly with an indentation – followed by a signature and some line ending. As already discussed, signatures and indentations are relatively fuzzy concepts, which means that the rules need to define a lot of cases and exceptions. There are also some template elements in the Wikipedia syntax that can change the interpretation of a comment, e.g. outdents[4].

Grammars are a theoretical concept that matches such a rule system very well. All different cases can be specified in detail and it is easy to build more complex rules out of multiple

[2] https://en.wikipedia.org/wiki/Wikipedia:Talk_page_guidelines, as seen on Feb. 14, 2017

[3] https://en.wikipedia.org/wiki/Help:Archiving_a_talk_page, as seen on Feb. 14, 2017

[4] https://en.wikipedia.org/wiki/Template:Outdent, as seen on Feb. 14, 2017

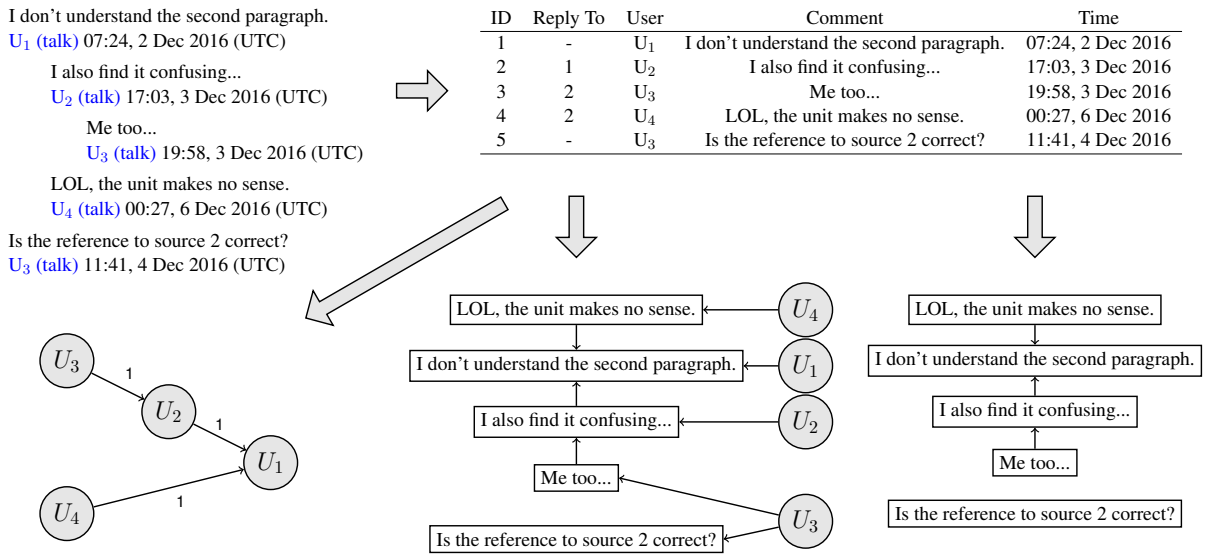


Figure 1: Schematic representation of the core parser component: A table (top right) is extracted from a talk page (top left) using a grammar-based rule set. It can then be transformed into various output formats, e.g. a one-mode (user- / comment-) graph (bottom left and right) or a two-mode graph (bottom center)

smaller ones. In particular we use Parsing Expression Grammars (Ford, 2004) which are very efficiently implemented in the Boost.Spirit library for C++. The library allows developers to essentially write the grammar in C++ syntax, which is then compiled – note the possibility for compile-time optimisation – to a highly optimised parser. Beside its efficient implementation, another advantage of using grammars is extensibility. Whenever we encountered mismatched or incorrectly parsed comments we added another case to our grammar and no real application logic had to be written.

After extracting the comments (with username, date, ...) from the raw talk page the core parser has various output formats including lists of the comments (formats: CSV, JSON, human-readable) and users as well as comment networks (formats: GML, GraphML, Graphviz) (cf. Figure 1).

3.2 Wiki Markup Crawler Component

The crawler component is responsible for obtaining the Wiki markup of a talk page using the Wikipedia website as a source. The program expects a text file containing the URLs to the Wikipedia articles as input. It then connects to the server to retrieve the HTML source code, from which we extract the relevant Wiki markup and which we finally feed to the core parser to get the structured talk page data for each article in the list of URLs.

Our crawler is also able to fetch archived pages and includes their content in the downloaded markup file. Finally, it also fetches transcluded content by searching for the respective Wiki-markup commands. If the transcluded content belongs to the namespace *Talk* – i.e. the name of the transcluded content starts with *Talk:* – it is part of a talk page and is therefore included. All other transcluded content is not included as it is unnecessary for the talk page analysis.

The crawler component should be used for small to medium-scale studies that rely on up-to-date talk page data. Setting up the crawler is very intuitive, but the need to contact the server may make it unfeasible to work with very large Wikipedia data.

3.3 Wiki Markup Dump Component

To be able to work with all data at once – without connecting to their server too often – Wikipedia offers a download of full database dumps that include e.g. all articles, all talk pages and all user pages.[5] For GraWiTas we also provide a program that is able to read such large XML dumps efficiently and feed talk pages to our core parser. Users can select if they want to parse all talk pages

[5] `https://en.wikipedia.org/wiki/Wikipedia:Database_download`, as seen on Feb. 14, 2017

in the dump or only some particular ones characterised by a list of article titles.

The dump component can be used for large-scale studies that look at all talk pages of Wikipedia at once. Compared to the crawler, obtaining the Wiki markdown is faster. However, this comes at the price that users have to download and maintain the large XML dump files.

4 Evaluation

We ran a very small evaluation study to assess the speed and accuracy of our core parser. For analysing the speed, we generated large artificial talk pages from smaller existing ones and measured the time our parser took. Using an Intel(R) Core(TM) i7 M 620 @ 2.67GHz processor, we obtained parsing times under 50ms for file sizes smaller than 1MB (∼2000 Comments). For files up to 10MB (∼20000 Comments) it took around 300ms. This leads to an overall parsing speed of around 30MB/s for average-sized comments. However, real world talk pages very rarely even pass the 1MB mark.

To evaluate accuracy, we picked a random article with a talk page and verified the extracted comments manually. Hereby, an accuracy of 0.97 was achieved. Although such a small evaluation is of course far from comprehensive, it shows that our parser is on the one hand fast enough to parse large numbers of talk pages and on the other hand yields a high enough accuracy for real-world applications.

5 Related Work

There have been previous attempts at parsing Wikipedia talk pages. Massa (2011) focused on user talk pages, where the conventions are different from article talk pages. Ferschke et al. (2012) rely on indentation for extracting relationships between comments, but use the revision history to identify the authors and comment creation dates. However, they do not offer any information on whether archived, transcluded and outdented content is handled correctly, and their implementation has not been made public. Laniado et al. (2011) infer the structure of discussions from indentation similar to our approach. They use user signatures to infer comment metadata (author and date). Their parser is available on Github[6]. However, it does not handle transcluded talk page content, nor the outdent template. Their parser outputs a CSV file with custom fields.

6 Conclusion

The possibility of parsing Wikipedia talk pages into networks provides many new avenues for NLP researchers interested in collaboration patterns. A usable parser is a prerequisite for this type of research. Unlike previous implementations, our parser correctly handles many quirks of Wikipedia talk page syntax, from archived and transcluded talk page contents to outdented comments. It can produce output in a number of standardised formats including GML as well as a custom text file format. Finally, it requires very little initial setup. The GraWiTas source code is publicly available[7], together with a web app demonstrating the core parser component.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media"

References

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the article: Recognizing dialog acts in wikipedia talk pages. In *Proc. of EACL'12*, pages 777–786, Stroudsburg, PA, USA. ACL.

Bryan Ford. 2004. Parsing expression grammars: A recognition-based syntactic foundation. *SIGPLAN Not.*, 39(1):111–122.

David Laniado, Riccardo Tasso, Yana Volkovich, and Andreas Kaltenbrunner. 2011. When the wikipedians talk: Network and tree structure of wikipedia discussion pages. In *Proc. of ICWSM, Barcelona, Spain, July 17-21, 2011*.

Jun Liu and Sudha Ram. 2011. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Trans. Manag. Inform. Syst.*, 2(2).

Paolo Massa. 2011. Social networks of wikipedia. In *HT'11, Proc. of ACM Hypertext 2011, Eindhoven, The Netherlands, June 6-9, 2011*, pages 221–230.

Fernanda B. Viegas, Martin Wattenberg, Jesse Kriss, and Frank Van Ham. 2007. Talk before you type: Coordination in wikipedia. In *Proc. of HICSS'07*. IEEE.

[6] `https://github.com/sdivad/WikiTalkParser`, as seen on Feb. 14, 2017

[7] `https://github.com/ace7k3/grawitas`

TWINE: A real-time system for TWeet analysis via INformation Extraction

Debora Nozza, Fausto Ristagno, Matteo Palmonari,
Elisabetta Fersini, Pikakshi Manchanda, Enza Messina
University of Milano-Bicocca / Milano
{debora.nozza, palmonari, fersini,
pikakshi.manchanda, messina}@disco.unimib.it
f.ristagno@campus.unimib.it

Abstract

In the recent years, the amount of user generated contents shared on the Web has significantly increased, especially in social media environment, e.g. Twitter, Facebook, Google+. This large quantity of data has generated the need of reactive and sophisticated systems for capturing and understanding the underlying information enclosed in them. In this paper we present TWINE, a **real-time** system for the **big data** analysis and **exploration** of information extracted from Twitter **streams**. The proposed system based on a Named Entity Recognition and Linking pipeline and a multi-dimensional spatial geo-localization is managed by a scalable and flexible architecture for an interactive visualization of micropost streams insights. The demo is available at *http://twine-mind.cloudapp.net/streaming*[1,2].

1 Introduction

The emergence of social media has provided new sources of information and an immediate communication medium for people from all walks of life (Kumar et al., 2014). In particular, Twitter is a popular microblogging service that is particularly focused on the speed and ease of publication. Everyday, nearly 300 million active users share over 500 million of posts[3], so-called tweets, principally using mobile devices.

Twitter has several advantages compared to traditional information channels, i.e. tweets are created in real-time, have a broad coverage over a wide variety of topics and include several useful embedded information, e.g. time, user profile and geo-coordinates if present.

Mining and extracting relevant information from this huge amount of microblog posts is an active research topic, generally called Information Extraction (IE). One of the key subtask of IE is Named Entity Recognition and Linking (NEEL), aimed to first identify and classify named entities such as people, locations, organisations and products, then to link the recognized entity mentions to a Knowledge Base (KB) (Derczynski et al., 2015).

Although several Information Extraction models have been proposed for dealing with microblog contents (Bontcheva et al., 2013; Derczynski et al., 2015), only few of them focused on the combination of these techniques with big data architecture and user interface in order to perform and explore real-time analysis of social media content streams. Moreover, the majority of these research studies are event-centric, in particular focusing on the tasks of situational awareness and event detection (Kumar et al., 2011; Leban et al., 2014; Sheth et al., 2014; Zhang et al., 2016).

In this paper we propose TWINE, a system that visualizes and efficiently performs real-time big data analytics on *user-driven* tweets via Information Extraction methods.

TWINE allows the user to:

- perform real-time monitoring of tweets related to their topics of interest, with unrestricted keywords;
- explore the information extracted by semantic-based analysis of large amount of tweets, i.e. (i) recognition of named entities and the information of the correspondent

[1] At the moment, the application is deployed on Azure client service with traffic and storage limits given by the provider.

[2] The TWINE system requires Twitter authentication, if you do not want to use your twitter account you can try the demo at *http://twine-mind.cloudapp.net/streaming-demo*.

[3] http://www.internetlivestats.com/

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 25–28

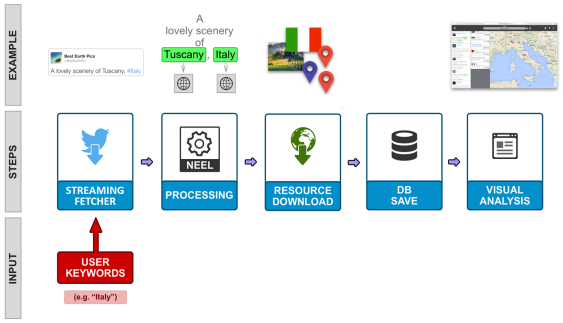


Figure 1: TWINE system overview.

KB resources, (ii) multi-dimensional spatial geo-tagging for each tweet, including the geo-localization of the named entities identified as locations and (iii) two semantic-driven interactive visualization interfaces.

The following section will present the details of the architecture for supporting real-time tweets analysis and the description of the conceived graphical user interface.

2 TWINE system

The proposed system TWINE, acronym for TWeet analysis via INformation Extraction, is a real-time system for the analysis and exploration of information extracted from Twitter data. Figure 1 outlines its macro steps coupled with corresponding examples.

Given a set of keywords provided by the user (e.g. "Italy") as input query, the system fetches the stream of all the tweets (text and tweet's author information) matching the keywords using Twitter APIs. Next, each tweet text is processed by the NEEL pipeline. This step provides as output a set of recognized named entities linked to the correspondent KB resource (e.g. *Tuscany* - *http://dbpedia.org/page/Tuscany*). After this elaboration, the system retrieves all the additional relevant information needed for the exploration: from the KB we extract the resource data, i.e. image, text description, type and coordinates if the entity is a location, and from Twitter we extract the tweet author account's profile location, that is resolved wih a georeferincing system.

This information are subsequently stored in a database that incrementally enriches information generated by the precedent phases. Then, the TWINE web interface fetches the stored data from the DB for populating two different interactive visualisation interfaces.

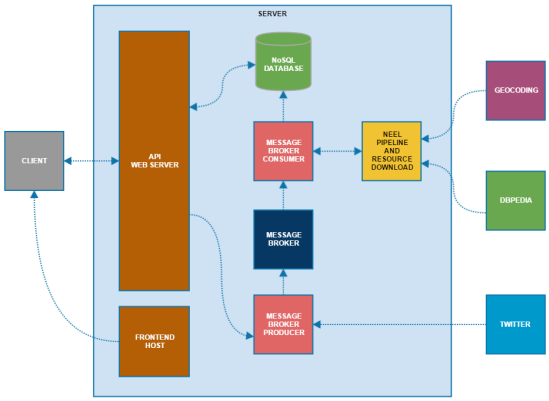


Figure 2: TWINE system architecture.

2.1 System Architecture

The proposed system is implemented using a centralized system architecture, as shown in Figure 2. The main requirement was to develop a system able to process in real-time large incoming of data streams.

In TWINE, all the aforementioned processes are triggered by the user from the client and elaborated on the server-side, i.e. the streaming fetching phase, the NEEL processing, the KB resources retrieval, the geo-codification of the locations and the database storing.

With this design implementation all the computations are performed on the server. This improves the independence on the client technical specifications, preventing different problems such as slow loading, high processor usage and even freezing.

The system architecture, presented in Figure 2, is composed of several independent modules:

External Services. The system makes use of Twitter APIs for fetching the streaming of tweets given an input query, a SPARQL endpoint over the DBpedia data set for the retrieval of the KB resource information and a georeferencing system, OpenStreetMap[4], to obtain the geographic coordinates from the tweet author account's profile location.

NEEL pipeline. This module uses the NEEL pipeline proposed by Caliano et al. (2016) on the tweets.

Message Broker system. This module is necessary to build pipelines for processing streaming data in real time, in such a way that components

[4]https://www.openstreetmap.org/

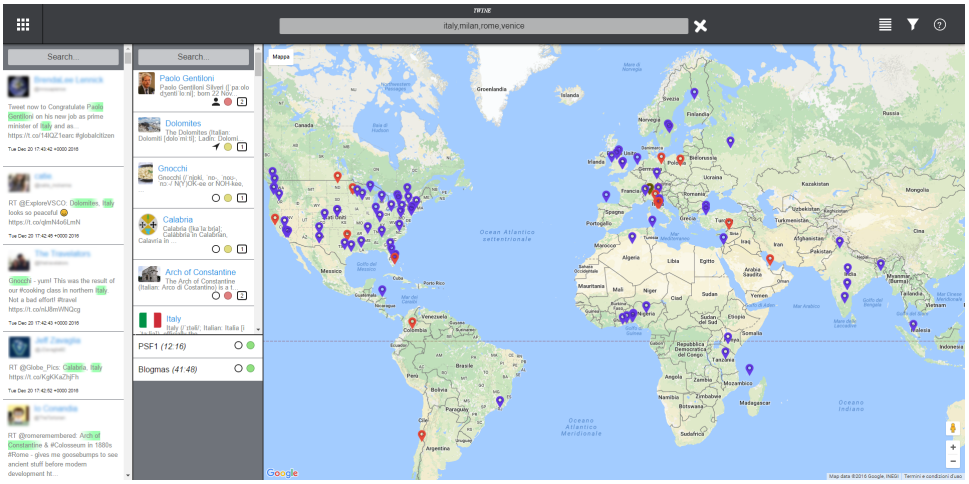


Figure 3: TWINE Map View snapshot.

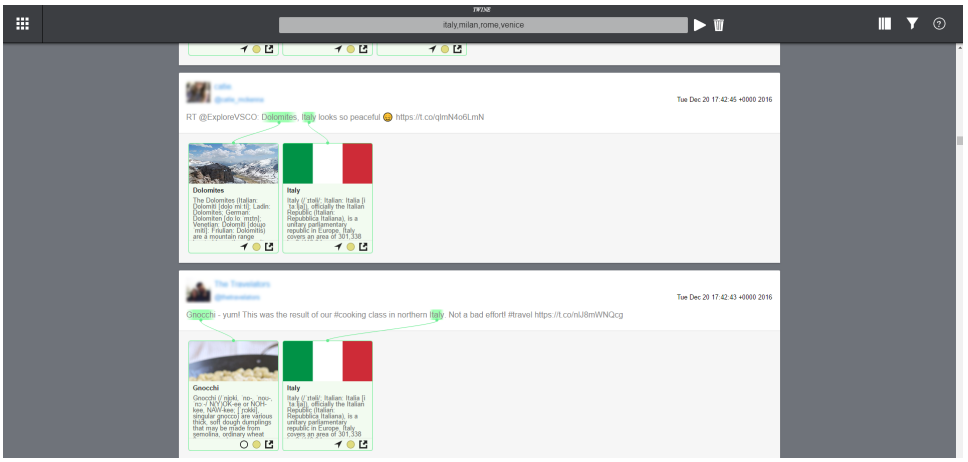


Figure 4: TWINE List View snapshot.

can exchange data reliably. The Apache Kafka platform[5] permits us to store and process the data in a fault-tolerant way and to ignore the latency due to the Information Extraction processing.

Database. All the source and processed data are stored in a NoSQL database. In particular, we choose a MongoDB[6] database because of its flexibility, horizontal scalability and its representation format that is particularly suitable for storing Twitter contents.

Frontend host and API web server. The presence of these two server-side modules is motivated by the need of make the TWINE user-interface independent on its functionalities. In this way, we improve the modularity and flexibility of the entire system.

[5]https://kafka.apache.org/
[6]http://www.mongodb.org/

2.2 User Interface

TWINE provides two different visualisations of the extracted information: the Map View, which shows the different geo-tags associated with tweets in addition to the NEEL output, and the List View, that better emphasizes the relation between the text and its named entities.

The Map View (Figure 3) provides in the top panel a textual search bar where users can insert keywords related to their topic of interest (e.g. *italy, milan, rome, venice*). The user can also, from left to right, start and stop the stream fetching process, clear the current results, change View and apply semantic filters related to the geo-localization and KB resource characteristics, i.e. type and classification confidence score.

Then, in the left-hand panel the user can read the content of each fetched tweet (text, user information and recognized named entities) and

directly open it in the Twitter platform.

The center panel can be further divided into two sub-panels: the top one shows the information about the Knowledge Base resources related to the linked named entities present in the tweets (image, textual description, type as symbol and the classification confidence score), and the bottom one provides the list of the recognized named entities for which it does not exist a correspondence in the KB, i.e. NIL entities.

These two panels, the one that reports the tweets and the one with the recognized and linked KB resources, are responsive. For example, by clicking on the entity *Italy* in the middle panel, only tweets containing the mention of the entity *Italy* will be shown in the left panel. Respectively, by clicking on a tweet, the center panel will show only the related entities.

In the right-hand panel, the user can visualize the geo-tag extracted from the tweets, (i) the original geo-location where the post is emitted (*green marker*), (ii) the user-defined location for the user account's profile (*blue marker*) and (iii) the geo-location of the named entities extracted from the tweets, if the corresponding KB resource has the latitude-longitude coordinates (*red marker*).

Finally, a text field is present at the top of the first two panels to filter the tweets and KB resources that match specific keywords.

The List View is reported in Figure 4. Differently from the Map View, the focus is on the link between the words, i.e. recognized named entities, and the corresponding KB resources. In the reported example, this visualisation is more intuitive for catching the meaning of *Dolomites* and *Gnocchi* thanks to a direct connection between the named entities and the snippet and the image of associated KB resources.

3 Conclusion

We introduced TWINE, a system that provides an efficient real-time data analytics platform on streaming of social media contents. The system is supported by a scalable and modular architecture and by an intuitive and interactive user interface.

As future work, we intend to implement a distributed solution in order to faster and easier manage huge quantity of data. Additionally, current integrated modules will be improved: the NEEL pipeline will be replaced by a multi-lingual and more accurate method, the web interface will include more insights such as the user network information, a heatmap visualization and a time control filter.

References

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. 2013. Twitie: An open-source information extraction pipeline for microblog text. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, pages 83–90.

Davide Caliano, Elisabetta Fersini, Pikakshi Manchanda, Matteo Palmonari, and Enza Messina. 2016. Unimib: Entity linking in tweets using jaro-winkler distance, popularity and coherence. In *Proceedings of the 6th International Workshop on Making Sense of Microposts (# Microposts)*.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49.

Shamanth Kumar, Geoffrey Barbier, Mohammad Ali Abbasi, and Huan Liu. 2011. Tweettracker: An analysis tool for humanitarian and disaster relief. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*.

Shamanth Kumar, Fred Morstatter, and Huan Liu. 2014. *Twitter data analytics*. Springer.

Gregor Leban, Blaz Fortuna, Janez Brank, and Marko Grobelnik. 2014. Event registry: learning about world events from news. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 107–110.

Amit Sheth, Ashutosh Jadhav, Pavan Kapanipathi, Chen Lu, Hemant Purohit, Gary Alan Smith, and Wenbo Wang. 2014. Twitris: A system for collective social intelligence. In *Encyclopedia of Social Network Analysis and Mining*, pages 2240–2253. Springer.

Xiubo Zhang, Stephen Kelly, and Khurshid Ahmad. 2016. The slandail monitor: Real-time processing and visualisation of social media data for emergency management. In *Proceedings of the 11th International Conference on Availability, Reliability and Security*, pages 786–791.

Alto: Rapid Prototyping for Parsing and Translation

Johannes Gontrum **Jonas Groschwitz*** **Alexander Koller*** **Christoph Teichmann***
University of Potsdam, Potsdam, Germany / * Saarland University, Saarbrücken, Germany
gontrum@uni-potsdam.de / *{groschwitz|koller|teichmann}@coli.uni-saarland.de

Abstract

We present *Alto*, a rapid prototyping tool for new grammar formalisms. Alto implements generic but efficient algorithms for parsing, translation, and training for a range of monolingual and synchronous grammar formalisms. It can easily be extended to new formalisms, which makes all of these algorithms immediately available for the new formalism.

1 Introduction

Whenever a new grammar formalism for natural language is developed, there is a prototyping phase in which a number of standard algorithms for the new formalism must be worked out and implemented. For monolingual grammar formalisms, such as (probabilistic) context-free grammar or tree-adjoining grammar, these include algorithms for chart parsing and parameter estimation. For synchronous grammar formalisms, we also want to decode inputs into outputs, and binarizing grammars becomes nontrivial. Implementing these algorithms requires considerable thought and effort for each new grammar formalism, and can lead to faulty or inefficient prototypes. At the same time, there is a clear sense that these algorithms work basically the same across many different grammar formalisms, and change only in specific details.

In this demo, we address this situation by introducing *Alto*, the Algebraic Language Toolkit. Alto is based on Interpreted Regular Tree Grammars (IRTGs; (Koller and Kuhlmann, 2011)), which separate the derivation process (described by probabilistic regular tree grammars) from the interpretation of a derivation tree into a value of the language. In this way, IRTGs can capture a wide variety of monolingual and synchronous grammar formalisms (see Fig. 1 for some examples). By selecting an appropriate algebra in which the values of the language are constructed, IRTGs can describe languages of objects that are not strings, including string-to-tree and tree-to-tree mappings, which have been used in machine translation, and synchronous hyperedge replacement grammars, which are being used in semantic parsing.

One advantage of IRTGs is that a variety of algorithms, including the ones listed above, can be expressed generically in terms of operations on regular tree grammars. These algorithms apply identically to all IRTG grammars and Alto offers optimized implementations. Only an algebra-specific decomposition operation is needed for each new algebra. Thus prototyping for a new grammar formalism amounts to implementing an appropriate algebra that captures new interpretation operations. All algorithms in Alto then become directly available for the new formalism, yielding an efficient prototype at a much reduced implementation effort.

Alto is open source and regularly adds new features. It is available via its website:
`https://bitbucket.org/tclup/alto`.

2 An example grammar

Let us look at an example to illustrate the Alto workflow. We will work with a synchronous string-to-graph grammar, which Alto's GUI displays as in Fig. 2. The first and second column describe a weighted regular tree grammar (wRTG, (Comon et al., 2007)), which specifies how to rewrite nonterminal symbols such as S and NP recursively in order to produce *derivation trees*. For instance, the tree shown in the leftmost panel of Fig. 3 can be derived using this grammar, starting with the start symbol S, and is assigned a weight (= probability) of 0.24. These derivation trees

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 29–32

Formalism	Reference	Algebra(s)
Context-Free Grammars (CFGs)	(Hopcroft and Ullman, 1979)	String
Hyperedge Replacement Grammars (HRGs)	(Chiang et al., 2013)	Graph
Tree Substitution Grammars	(Sima'an et al., 1994)	String / Tree
Tree-Adjoining Grammars (TAGs)	(Joshi et al., 1975)	TAG string / TAG tree
Synchronous CFGs	(Chiang, 2007)	String / String
Synchronous HRGs	(Chiang et al., 2013)	String / Graph
String to Tree Transducer	(Galley et al., 2004)	String / Tree
Tree to Tree Transducer	(Graehl et al., 2008)	Tree / Tree

Figure 1: Some grammar formalisms that Alto can work with.

serve as abstract syntactic representations, along the lines of derivation trees in TAG.

Next, notice the column "english" in Fig. 2. This column describes how to *interpret* the derivation tree in an interpretation called "english". It first specifies a *tree homomorphism*, which maps derivation trees into terms over some algebra by applying certain rewrite rules bottom-up. For instance, the "boy" node in the example derivation tree is mapped to the term $t_1 = *(\mathsf{the}, \mathsf{boy})$. The entire subtree then maps to a term of the form $*(?1, *(\mathsf{wants}, *(\mathsf{to}, ?2)))$, as specified by the row for "wants2" in the grammar, where $?1$ is replaced by t_1 and $?2$ is replaced by the analogous term for "go". The result is shown at the bottom of the middle panel in Fig. 3. Finally, this term is *evaluated* in the underlying algebra; in this case, a simple string algebra, which interprets the symbol $*$ as string concatenation. Thus the term in the middle panel evaluates to the string "the boy wants to go", shown in the "value" field.

The example IRTG also contains a column called "semantics". This column describes a second interpretation of the derivation tree, this time into an algebra of graphs. Because the graph algebra is more complex than the string algebra, the function symbols look more complicated. However, the general approach is exactly the same as before: the grammar specifies how to map the derivation tree into a term (bottom of rightmost panel in Fig. 3), and then this term is evaluated in the respective algebra (here, the graph shown at the top of the rightmost panel).

Thus the example grammar is a synchronous grammar which describes a relation between strings and graphs. When we parse an input string w, we compute a *parse chart* that describes all grammatically correct derivation trees that interpret to this input string. We do this by computing a *decomposition grammar* for w, which describes all terms over the string algebra that evaluate to w; this step is algebra-specific. From this, we calculate a regular tree grammar for all derivation trees that the homomorphism maps into such a term, and then intersect it with the wRTG. These operations can be phrased in terms of generic operations on RTGs; implementing these efficiently is a challenge which we have tackled in Alto. We can compute the best derivation tree from the chart, and map it into an output graph. Similarly, we can also decode an input graph into an output string.

3 Algorithms in Alto

Alto can read IRTG grammars and corpora from files, and implements a number of core algorithms, including: automatic binarization of monolingual and synchronous grammars (Büchse et al., 2013); computation of parse charts for given input objects; computing the best derivation tree; computing the k-best derivation trees, along the lines of (Huang and Chiang, 2005); and decoding the best derivation tree(s) into output interpretations. Alto supports PCFG-style probability models with both maximum likelihood and expectation maximization estimation. Log-linear probability models are also available, and can be trained with maximum likelihood estimation. All of these functions are available through command-line tools, a Java API, and a GUI, seen in Fig. 2 and 3.

We have invested considerable effort into making these algorithms efficient enough for practical use. In particular, many algorithms for wRTGs in Alto are implemented in a lazy fashion, i.e. the rules of the wRTG are only calculated by need; see e.g. (Groschwitz et al., 2015; Groschwitz et al., 2016). Obviously, Alto cannot be as efficient for well-established tasks like PCFG parsing

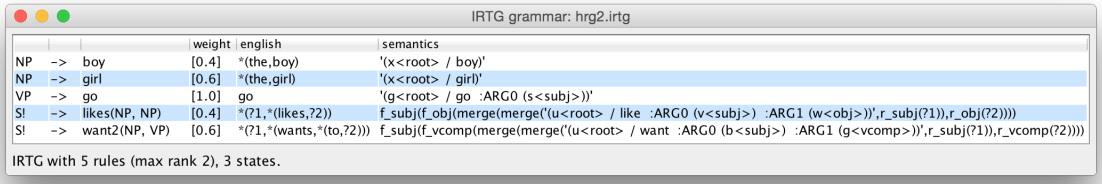


Figure 2: An example IRTG with an English and a semantic interpretation (Alto screenshot).

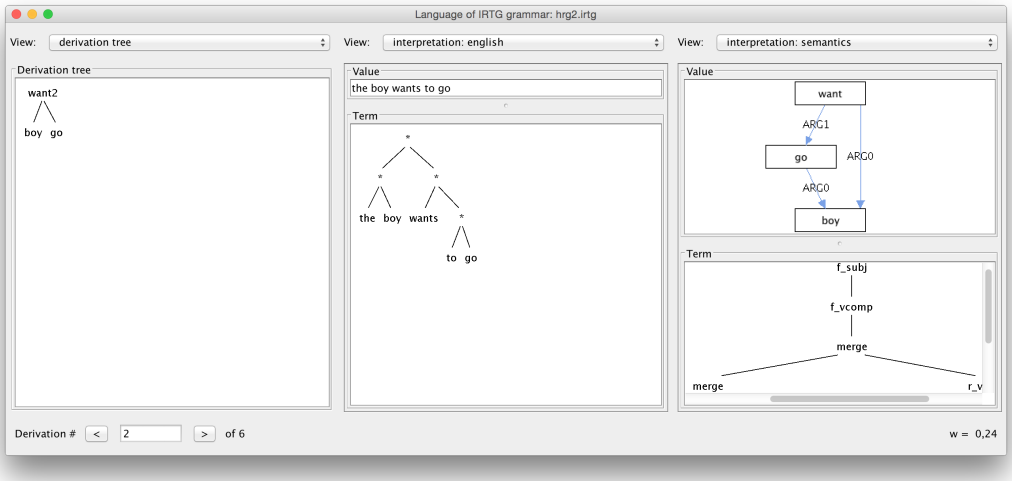


Figure 3: A derivation tree with interpreted values (Alto screenshot).

as a parser that was implemented and optimized for this specific grammar formalism. Nonetheless, Alto is fast enough for practical use with treebank-scale gramars, and for less mainstream grammar formalisms can be faster than specialized implementations for these formalisms. For instance, Alto is the fastest published parser for Hyperedge Replacement Grammars (Groschwitz et al., 2015). Alto contains multiple algorithms for computing the intersection and inverse homomorphism of RTGs, and a user can choose the combination that works best for their particular grammar formalism (Groschwitz et al., 2016).

The most recent version adds further performance improvements through the use of a number of pruning techniques, including coarse-to-fine parsing (Charniak et al., 2006). With these, Section 23 of the WSJ corpus can be parsed in a couple of minutes.

4 Extending Alto

As explained above, Alto can capture any grammar formalism whose derivation trees can be described with a wRTG, by interpreting these into different algebras. For instance, the difference between Context-Free and Tree-Adjoining Grammars in Alto is that CFGs use the simple string algebra outlined in Section 2, whereas for TAG we use a special "TAG string algebra" which defines string wrapping operations (Koller and Kuhlmann, 2012). All algorithms mentioned in Section 3 are generic and do not make any assumptions about what algebras are being used. As explained above, the only algebra-specific step is to compute decomposition grammars for input objects.

In order to implement a new algebra, a user of Alto simply derives a class from the abstract base class `Algebra`, which amounts to specifying the possible values of the algebra (as a Java class) and implementing the operations of the algebra as Java methods. If Alto is also to parse objects from this algebra, the class needs to implement a method for computing decomposition grammars for the algebra's values. Alto comes with a number of algebras built in, including string algebras for Context-Free and Tree-Adjoining grammars as well as tree, set, and graph algebras. All of these can be used in parsing. By parsing sets in a set-to-string grammar for example, Alto can generate referring expressions (Engonopoulos and Koller, 2014).

Finally, Alto has a flexible system of *input and output codecs*, which can map grammars and algebra values to string representations and vice versa. A key use of these codecs is reading grammars in native input format and converting them into IRTGs. Users can provide their own codecs to maximize interoperability with existing code.

5 Conclusion and Future Work

Alto is a flexible tool for the rapid implementation of new formalisms. This flexibility is based on a division of concerns between the generation and the interpretation of grammatical derivations. We hope that the research community will use Alto on newly developed formalisms and on novel combinations for existing algebras. Further, the toolkit's focus on balancing generality with efficiency supports research using larger datasets and grammars.

In the future, we will implement algorithms for the automatic induction of IRTG grammars from corpora, e.g. string-to-graph corpora, such as the AMRBank, for semantic parsing (Banarescu et al., 2013). This will simplify the prototyping process for new formalisms even further, by making large-scale grammars for them available more quickly. Furthermore, we will explore ways for incorporating neural methods into Alto, e.g. in terms of supertagging (Lewis et al., 2016).

Acknowledgements

This work was supported by DFG grant KO 2916/2-1.

References

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop*.

M. Büchse, A. Koller, and H. Vogler. 2013. Generic binarization for parsing and translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

E. Charniak, M. Johnson, M. Elsner, J. Austerweil, D. Ellis, I. Haxton, C. Hill, R. Shrivaths, J. Moore, M. Pozar, and T. Vu. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*.

D. Chiang, J. Andreas, D. Bauer, K. M. Hermann, B. Jones, and K. Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, and C. Löding. 2007. *Tree Automata Techniques and Applications*. published online - `http://tata.gforge.inria.fr/`.

N. Engonopoulos and A. Koller. 2014. Generating effective referring expressions using charts. In *Proceedings of the INLG and SIGDIAL 2014*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *HLT-NAACL 2004: Main Proceedings*.

J. Graehl, K. Knight, and J. May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

J. Groschwitz, A. Koller, and C. Teichmann. 2015. Graph parsing with s-graph grammars. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

J. Groschwitz, A. Koller, and M. Johnson. 2016. Efficient techniques for parsing with tree automata. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

J. E. Hopcroft and J. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*.

A. K. Joshi, L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

A. Koller and M. Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*.

A. Koller and M. Kuhlmann. 2012. Decomposing tag algorithms using simple algebraizations. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*.

M. Lewis, K. Lee, and L. Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of NAACL-HLT*.

K. Sima'an, R. Bod, S. Krauwer, and R. Scha. 1994. Efficient disambiguation by means of stochastic tree substitution grammars. In *Proceedings of International Conference on New Methods in Language Processing*.

CASSANDRA: A multipurpose configurable voice-enabled human-computer-interface

Tiberiu Boros, Stefan Daniel Dumitrescu and Sonia Pipa
Research Center for Artificial Intelligence
Romanian Academy
Bucharest, Romania
tibi@racai.ro, sdumitrescu@racai.ro, sonia@racai.ro

Abstract

Voice enabled human computer interfaces (HCI) that integrate automatic speech recognition, text-to-speech synthesis and natural language understanding have become a commodity, introduced by the immersion of smart phones and other gadgets in our daily lives. Smart assistants are able to respond to simple queries (similar to text-based question-answering systems), perform simple tasks (call a number, reject a call etc.) and help organizing appointments. With this paper we introduce a newly created process automation platform that enables the user to control applications and home appliances and to query the system for information using a natural voice interface. We offer an overview of the technologies that enabled us to construct our system and we present different usage scenarios in home and office environments.

1 Introduction

Major mobile developers are currently including some form of personal assistants in their operating systems, which enable users to control their device using natural voice queries (see Google Now in Android, Siri in Apple's iOS, Microsoft's Cortana, etc.). While these assistants offer powerful integration with the device, they are restricted to that device only - the user (a) is able to control its device, (b) has access to information from simple queries (QA systems included by major competitors in the mobile OS world are able to understand and automatically summarize answers to queries such as: how is the weather today?, who is the president of the United States or who was Michael Jackson) and (c) can organize his/her agenda according to the documents stored in his/her own cloud-hosted storage (Google is able to parse and obtain information from plane tickets and bookings and automatically provides calendar entries as well as general tips such as your plane leaves tomorrow at 11 AM and you should be at the airport before 10 AM due to traffic). We introduce our natural-voice assistive system for process automation that enables control of external devices/gadgets as well as allowing the user to interact with the system in the form of information queries, much like a personal assistant existing now on mobile devices. The user interacts with the system using his/her voice, the system understands and acts accordingly by controlling external devices or by responding to the user with the requested information. All the work presented in this paper was done during the implementation of the Assistive Natural-language, Voice-controlled System for Intelligent Buildings (ANVSIB) national project. We cover aspects related to system architecture, challenges involved by individual tasks, performance figures of the sub-modules and technical decisions related to the development of a working prototype. The tools and technologies are divided in 3 main topics: (a) automatic speech recognition (ASR); (b) text-to-speech synthesis (TTS) and (c) integration with home automation services.

2 System architecture

From a logical point of view, the system is divided in three components (each presented in this paper): ASR, TTS, and integration with external services using natural language understanding (NLU). From an procedural point of view, the system has two distinct entities: one entity acts as an endpoint(client) and is implemented as an android application that is responsible for (a) acquisition of

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 33–36

speech data from the user, processing and recognition (all using external services), as well as the (b) presentation of data to the user. The second entity acts as a server and is responsible for receiving text-input from the endpoint, identifying a scenario from a limited set, extracting parameters, performing the necessary operations and returning information to the endpoint in the form of text, images and sound.

Because we wanted to keep our system as scalable as possible, Automatic Speech Recognition and Text-to-speech synthesis operations are carried out on external servers. The endpoint can be configured to access both Cassandra's own ASR and TTS services (described here) as well as the ASR and TTS servers provided by Google Speech API.

2.1 Cassandra's TTS synthesis system

Text-to-speech synthesis refers to the conversion of any arbitrary (unrestricted) text into audio signal. The unrestricted requirement makes this task difficult and, while state-of-the-art systems produce remarkable results in terms of intelligibility and naturalness, the recipe for producing synthetic voices which are indistinguishable from natural ones has not yet been found. This limitation is caused by the fact that natural language understanding still poses serious challenges for machines and the fact that the surface form of the text does not provide sufficient cues for the prosodic realization of a spontaneous and expressive voice (Taylor, 2009).

TTS synthesis involves two major steps: (a) extraction of features from text and (b) conversion of symbolic representations into actual speech. The text processing step (step a) is usually composed of low-level text-processing tasks such as part-of-speech tagging, lemmatization, chunking, letter-to-sound conversion, syllabification etc. The signal processing task (step b) consists of selecting an optimal set of speech parameters (given the features provided by step a) and generating an acoustic signal that best fits these parameters. Text processing is performed by our in-house developed natural language processing pipeline called Modular Language Processing for Lightweight Applications (MLPLA) (Zafiu et al., 2015). Most of the individual modules have been thoroughly described in our previous work (Boros, 2013; Boros and Dumitrescu, 2015), so we only briefly list them here: the part-of-speech tagger is a neural inspired approach (Boros et al., 2013b) achieving 98.21% accuracy on the "1984" novel by G. Orwell using morphosyntactic descriptors (MSDs) (Erjavec, 2004) in the tag-set; all the lexical processing modules were described in (Boros, 2013)

For syllabification we used the onset-nucleus-coda (ONC) tagging strategy proposed in (Bartlett et al., 2009) and chunking is performed using a POS-based grammar described in (Ion, 2007).

Our TTS implements both unit-selection (Boros et al., 2013a) and statistical parametric speech synthesis. For Cassandra we chose to use parametric synthesis with our implementation of the STRAIGHT filter (Kawahara et al., 1999).

Our TTS system primarily supports English, German, French and Romanian and can be accessed for demonstration purposes from our web portal(link will be provided after evaluation). For other languages Cassandra uses Google TTS, which provides statistical parametric speech synthesis for a large number of languages.

2.2 Cassandra's speech recognition

Cassandra's Automatic Speech Recognition (ASR) service was created by our partners, the Speech and Dialogue Research Laboratory (http://speed.pub.ro) and it is a scalable and extensible on-line Speech-to-Text (S2T) solution. A demo version of the Speech-to-Text (S2T) system resides in the cloud or in SpeeDs IT infrastructure and can be accessed on-line using the Web API or a proprietary protocol. Client applications can be developed using any technology that is able to communicate through TCP-IP sockets with the server application. The server application can communicate with several clients, serving them either simultaneously or sequentially.

The speech-to-text system can be configured to transcribe different types of speech, from a number of domains and languages. It can be configured to instantiate multiple speech recognition engines (S2T Transcribers), each of these engines being responsible for transcribing speech from a specific domain (for example, TV news in Romanian, medical-related speech in Romanian, country names in English, etc.). The speech recognition engines are based on the open-source CMU Sphinx speech recognition toolkit. The ASR systems with small vocabulary and grammar language models were evaluated in depth in (Cucu

et al., 2015), while the ASR system with large vocabulary and statistical language model was evaluated in depth in (Cucu et al., 2014). The first ones have word error rates between 0 and 5%, while the large vocabulary ASR has a word error rate of about 16%.

2.3 Natural language processing

The core of Cassandra is driven by a natural language processing system that is responsible for receiving a text and, after analysis and parameter extraction, acting based on a predefined scenario. A scenario is the equivalent of a frame in frame-based dialogue systems. At any time, the end-user is able to alter Cassandra's configuration by creating scenarios or editing existing ones. A scenario is defined by its name and 3 components:

Component 1 contains a list of example sentences with parameters preceded by a special character '$' which is used to identify them. (e.g. "Set the temperature on $value degrees.")

Component 2 tells the system what to actions to take, depending on the scenario. Actions are described in a JSON with the following structure: (a) gadget - a unique identifier telling the system what module must be used for processing (e.g. a light, an A/C unit, the security system, etc.); (b) gadget_parameters - a free-form JSON structure that tells the system what parameters to pass on to the gadget (e.g. turn the lights on=1 or off=0, set A/C to X degrees, etc.). The values of these parameters can be either constants, predefined system variables or actual values extracted from the text. A gadget can return a text response.

Component 3 is used for feedback. After processing, this JSON is returned to the end-point which initiated the session and it is used to relay information back to the user. This is also a JSON with 2 attributes: (a) friendly_response - a text response that if present will be synthesized and played back to the user; the friendly response can include system variables, parameters extracted from the text or the response returned by the gadget; (b) launch_intent - a structure that informs the end-point that it must launch an external Android Intent with a given set of parameters; Cassandra comes with a predefined set of Intents for image preview, audio playback or video playback.

The methodology for scenario identification and parameter extraction is performed in three sequential steps: (a) the scenario is identified using Long-Short Term Memory neural networks and word embeddings; (b) parameter start/stop markers are then added using a deep neural network classifier trained on a window of 4 words; (c) next, parameter types are added using a window of six words, in which the parameters are ignored. Theoretically, parameter identification (c) and boundary detection (b) could be carried out simultaneously. However, we found that splitting this task in two steps works significantly better, mainly because it mitigates the data sparsity issue. An interesting observation is that using word embeddings in the scenario identification step, allowed the system to identify the query "It's too hot" as an air conditioning activity, though the training data only contained examples such as: "Set AC to $value", "Set the temperature to $value degrees" etc.

Because we wanted to keep the system as language independent as possible, the only external resources required for language adaptation is a word embeddings file extracted using word2vec (Mikolov et al., 2014). The process is simple and given a large enough corpus (e.g. a Wikipedia dump) one can obtain good embeddings by running the word2vec tool on the tokenized text.

3 Standard gadgets and usage scenarios

At submission time we have already implemented 4 demo scenarios:

Scenario 1 is a standard query system, in which the user can ask Cassandra various questions (for example "how is the weather") to which the system will respond using a Knowledge Base (KB). The KB was built using Wikipedia for Romanian and English.

Scenario 2 is a home automation system that allows the user to control home appliances using a natural voice interface. There are several predefined tasks such as multimedia, lighting, climate and security system control. Communication with these devices is performed through KNX (EN 50090, ISO/IEC 14543), which is a purpose-built standardized network communication protocol that is simple and scalable.

Scenario 3 is a set of hard-coded short questions/answers that increase Cassandra's appeal. For example, this Q/A set contains a game called "shrink". It enables the system to perform word associations in a similar manner in which a psychologist would ask a patient to do. This particular game proved to be very appealing to the users in

our test group primarily because it made the system seem more "life-like".

Scenario 4 is a business oriented demo. We integrated Cassandra with a Document Management System (DMS) and created an application that filters documents based on associated meta-data. The interaction is described using a large JSON in which all parameters are optional, missing parameters being ignored by the system. By doing so we are able to respond to queries such as: "Give me all invoices", "Give me all invoices issued to Acme Computers", "Give me all invoices issued to Acme computers that are due in two weeks" etc. We find this scenario particularly interesting from a commercial point-of-view, especially because it enables users access to the DMS without being forced to master any specific search skills. In a similar way, an application could be built to give users access to databases and enable them to construct complex queries and reports with no SQL knowledge whatsoever: "Give me a list of customers that bought tablets over the last 6 months and group them by age".

4 Conclusions and future development

Cassandra is an open-source, freely available personal assistant and, from our knowledge, the only system that is extensible to several languages, not only English, using minimal effort. We intend to further develop this system and extend the basic set of applications to suit most common usage scenarios, as well as to offer more complex NLP-powered business scenarios that integrate with various existing software and hardware implementations. A necessary next step in the near future is to create an open source repository that will enable the creation of a community of developers around it.

Acknowledgements: This work was supported by UEFISCDI, under grant PN-II-PT-PCCA-2013-4-0789, project Assistive Natural-language, Voice-controlled System for Intelligent Buildings (2013-2017).

References

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2009. On the syllabification of phonemes. In *Proceedings of Human Language Technologies: North American Chapter of the Association for Computational Linguistics*, pages 308–316. Association for Computational Linguistics.

Tiberiu Boros and Stefan Daniel Dumitrescu. 2015. Robust deep-learning models for text-to-speech synthesis support on embedded devices. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, pages 98–102. ACM.

Tiberiu Boros, Radu Ion, and tefan Daniel Dumitrescu. 2013a. The racai text-to-speech synthesis system.

Tiberiu Boros, Radu Ion, and Dan Tufis. 2013b. Large tagset labeling using feed forward neural networks. case study on romanian language. In *ACL (1)*, pages 692–700.

Tiberiu Boros. 2013. A unified lexical processing framework based on the margin infused relaxed algorithm. a case study on the romanian language. In *RANLP*, pages 91–97.

Horia Cucu, Andi Buzo, Lucian Petrică, Dragoş Burileanu, and Corneliu Burileanu. 2014. Recent improvements of the speed romanian lvcsr system. In *Communications (COMM), 2014 10th International Conference on*, pages 1–4. IEEE.

Horia Cucu, Andi Buzo, and Corneliu Burileanu. 2015. The speed grammar-based asr system for the romanian language. *ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY*, 18(1):33–53.

Tomaz Erjavec. 2004. Multext-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *LREC*.

Radu Ion. 2007. Word sense disambiguation methods applied to english and romanian. *PhD thesis. Romanian Academy, Bucharest*.

Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. 1999. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3):187–207.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2014. word2vec.

Paul Taylor. 2009. *Text-to-speech synthesis*. Cambridge university press.

Adrian Zafiu, Tiberiu Boros, and Stefan Daniel Dumitrescu. 2015. Modular language processing for lightweight applications. In *Proceedings of Language & Technology Conference*.

An Extensible Framework for Verification of Numerical Claims

James Thorne
Department of Computer Science
University of Sheffield, UK
j.thorne@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

In this paper we present our automated fact checking system demonstration which we developed in order to participate in the Fast and Furious Fact Check challenge. We focused on simple numerical claims such as "population of Germany in 2015 was 80 million" which comprised a quarter of the test instances in the challenge, achieving 68% accuracy. Our system extends previous work on semantic parsing and claim identification to handle temporal expressions and knowledge bases consisting of multiple tables, while relying solely on automatically generated training data. We demonstrate the extensible nature of our system by evaluating it on relations used in previous work. We make our system publicly available so that it can be used and extended by the community.[1]

1 Introduction

Fact checking is the task of assessing the truthfulness in spoken or written language. We are motivated by calls to provide tools to support journalists with resources to verify content at source (Cohen et al., 2011) or upon distribution. Manual verification can be too slow to verify information given the speed at which claims travel on social networks (Hassan et al., 2015a).

In the context of natural language processing research, the task of automated fact checking was discussed by Vlachos and Riedel (2014). Given a claim, a system for this task must determine what information is needed to support or refute the claim, retrieve the information from a knowledge base (KB) and then compute a deduction to assign a verdict. For example, in the claim of Figure 1 a system needs to recognize the named entity (Germany), the statistical property (population) and the year, link them to appropriate elements in a KB, and deduce the truthfulness of the claim using the absolute percentage error.

Input: Around 80 million people were inhabitants of Germany in 2015.
Data Source: `data/worldbank_wdi.csv`
Property: `population(Germany, 2015)`
Value: `81413145`
Absolute Percentage Error: 1.7%
Verdict: `TRUE`

Figure 1: Fact checking a claim by matching it to an entry in the knowldge base.

We contrast this task against rumour detection (Qazvinian et al., 2011) – a similar prediction task based on language subjectivity and growth of readership through a social network. While these are important factors to consider, a sentence can be true or false regardless of whether it is a rumour (Lukasik et al., 2016).

Existing fact checking systems are capable of detecting fact-check-worthy claims in text (Hassan et al., 2015b), returning semantically similar textual claims (Walenz et al., 2014); and scoring the truth of triples on a knowledge graph through semantic distance (Ciampaglia et al., 2015). However, neither of these are suitable for fact checking a claim made in natural language against a database. Previous works appropriate for this task operate on a limited domain and are not able to incorporate temporal information when checking time-dependent claims (Vlachos and Riedel, 2015).

In this paper we introduce our fact checking tool, describe its architecture and design decisions, evaluate its accuracy and discuss future work. We

[1] https://github.com/sheffieldnlp/numerical-fact-checking-eacl2017

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 37–40

highlight the ease of incorporating new information sources to fact check, which may be unavailable during training. To validate the extensibility of the system, we complete an additional evaluation of the system using claims taken from Vlachos and Riedel (2015). We make the source code publicly available to the community.

2 Design Considerations

We developed our fact-checking approach in the context of the HeroX challenge[2] – a competition organised by the fact checking organization FullFact[3]. The types of claims the system presented can fact check was restricted to those which require looking up a value in a KB, similar to the one in Figure 1. To learn a model to perform the KB look up (essentially a semantic parsing task), we extend the work of Vlachos and Riedel (2015) who used distant supervision (Mintz et al., 2009) to generate training data, obviating the need for manual labeling. In particular, we extend it to handle simple temporal expressions in order to fact check time-dependent claims appropriately, i. e. population in 2015. While the recently proposed semantic parser of Pasupat and Liang (2015) is also able to handle temporal expressions, it makes the assumption that the table against which the claim needs to be interpreted is known, which is unrealistic in the context of fact checking.

Furthermore, the system we propose can predict relations from the KB on which the semantic parser has not been trained, a paradigm referred to as zero-shot learning (Larochelle et al., 2008). We achieve this by learning a binary classifier that assesses how well the claim "matches" each relation in the KB. Finally, another consideration in our design is algorithmic accountability (Diakopoulos, 2016) so that the predictions and the decision process used by the system are interpretable by a human.

3 System Overview

Given an unverified statement, the objective of this system is to identify a KB entry to support or refute the claim. Our KB consists of a set of un-normalised tables that have been translated into simple Entity-Predicate-Value triples through a simple set of rules. In what follows we first describe the fact checking process used during testing (Section 3.1) and then how the relation matching module is trained and the features used for this purpose (Section 3.2).

[2] http://herox.com/factcheck

[3] https://fullfact.org

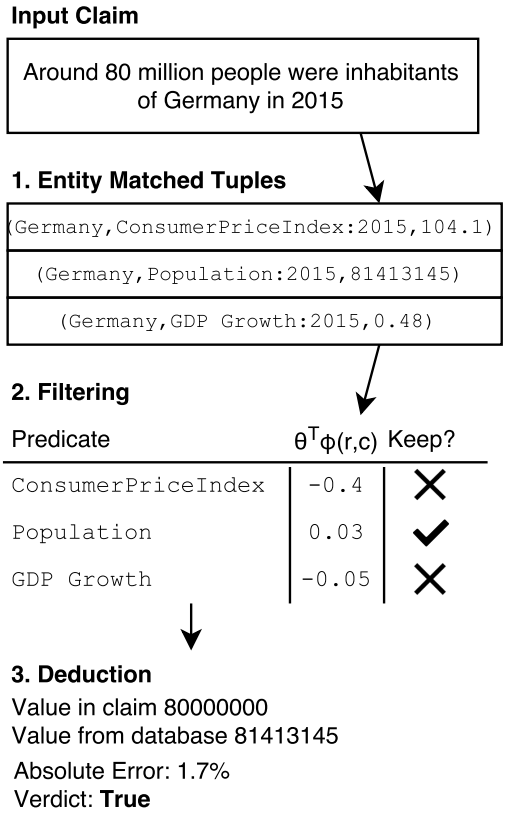


Figure 2: Relation matching step and filtering

3.1 Fact Checking

In our implementation, fact checking is a three step process, illustrated in Figure 2. Firstly, we link named entities in the claim to entities in our KB and retrieve a set of tuples involving the entities found. Secondly, these entries are filtered in a relation matching step. Using the text in the claim and the predicate as features, we classify whether this tuple is relevant. And finally, the values in the matched triples from the KB are compared to the value in the statement to deduce the verdict as follows: if there is at least one value with absolute percentage error lower than a threshold defined by the user, the claim is labeled true, otherwise false.

We model the relation matching as a binary classification task using logistic regression implemented in `scikit-learn`, predicting whether a predicate is a match to the given input claim. The aim of this step is to retain tuples for which the predicate in the Entity-Predicate-Value tuple can be described by the surface forms present in the

claim (positive-class) and discard the remainder (negative-class). We chose not to model this as a multi-class classification task (one class per predicate) to improve the extensibility of the system. A multi-class classifier requires training instances for every class, thus would not be applicable to predicates not seen during training. Instead, our aim is to predict the compatibility of a predicate w. r. t. the input claim.

For each of the candidate tuples $r_i \in R$, a feature vector is generated using lexical and syntactic features present in the claim and relation: $\phi(r_i, c)$. This feature vector is inputted to a logistic regression binary classifier and we retain all tuples where $\theta^T \phi(r_i, c) \geq 0$.

3.2 Training Data Generation

The training data for relation matching is generated using distant supervision and the Bing search engine. We first read the table and apply a set of simple rules to extract subject, predicate, object tuples, and for each named entity and numeric value we generate a query containing the entity name and the predicate. For example, the entry `(Germany,Population:2015,81413145)` is converted to the query *"Germany" Population 2015*. The queries are then executed on the Bing search engine and the top 50 web-page results are retained. We extract the text from the webpages using a script built around the BeautifulSoup package.[4] This text is parsed and annotated with co-reference chains using the Stanford CoreNLP pipeline (Manning et al., 2014).

Each sentence containing a mention of an entity and a number is used to generate a training example. The examples are labeled as follows. If the absolute percentage error between the value in the KB and the number extracted from text is below a threshold (an adjustable hyperparameter), the training instance is marked as a positive instance. Sentences which contain a number outside of this threshold are marked as negative instances. We make an exception for numbers tagged as dates where an exact match is required.

For each claim, the feature generation function, $\phi(r, c)$, outputs lexical and syntactic features from the claim and a set of custom indicator variables. Additionally, we include a bias feature for every unique predicate in the KB. For our lexical and syntactic features, we consider the words in the span between the entity and number as well as the dependency path between the entity and the number. To generalise to unseen relations, we include the ability to add custom indicator functions; for our demonstration we include a simple boolean feature indicating whether the intersection of the words in the predicate name and the sentence is not empty.

4 Evaluation

The system was field tested in the HeroX fact checking challenge - 40 general-domain claims chosen by journalists. Given the design of our system, we were restricted to claims that can only be answered by a KB look up (11 claims in total), returning the correct truth assessment for 7.5 of them according to the human judges. The half-correct one was due to not providing a fully correct explanation. To fact check a statement, the user only needs to enter the claim into a text box. The only requirement is to provide the system with an appropriate KB. Thus we ensured that the system can readily incorporate new tables taken from encyclopedic sources such as Wikipedia and the World Bank. In our system, this step is achieved by simply importing a CSV file and running a script to generate the new instances to train the relation matching classifier.

Analysis of our entry to this competition showed that two errors were caused by incorrect initial source data and one partial error caused by recalling a correct property but making an incorrect deduction. Of numerical claims that we did not attempt, we observed that many required looking up multiple entries and performing a more complex deduction step which was beyond the scope of this project.

We further validate the system by evaluating the ability of this fact checking system to make veracity assessments on simple numerical claims from the data set collected by (Vlachos and Riedel, 2015). Of the 4,255 claims about numerical properties about countries and geographical areas in this data set, our KB contained information to fact check 3,418. The system presented recalled KB entries for 3,045 claims (89.1%). We observed that the system was consistently unable to fact check two properties (undernourishment and renewable freshwater per capita). Analysis of these failure cases revealed too great a lexical difference

[4] `https://www.crummy.com/software/BeautifulSoup/`

between the test claims and the training data our system generated; the claims in the test cases were comparative in nature (e. g. country X has higher rate of undernourishment than country Y) whereas the training data generated using the method described in Section 3.2 are absolute claims.

A high number of false positive matches were generated, e. g. for a claim about population, other irrelevant properties were also recalled in addition. For the 3,045 matched claims, 17,770 properties were matched from the KB that had a score greater than or equal to the logistic score of the correct property. This means that for every claim, there were, on average, 5.85 incorrect properties also extracted from the KB. In our case, this did not yield false positive assignment of truth labels to the claims. This was because the absolute percentage error between the incorrectly retrieved properties and the claimed value was outside of the threshold we defined; thus these incorrectly retrieved properties never resulted in a true verdict, allowing the correct one to determine the verdict.

5 Conclusions and Future Work

The core capability of the system demonstration we presented is to fact check natural language claims against relations stored in a KB. Although the range of claims is limited, the system is a field-tested prototype and has been evaluated on a published data set (Vlachos and Riedel, 2015) and on real-world claims presented as part of the HeroX fact checking challenge. In future work, we will extend the semantic parsing technique used and apply our system to more complex claim types. Additionally, further work is required to reduce the number of candidate relations recalled from the KB. While this was not an issue in our case, we believe that ameliorating this issue will enhance the ability of the system to assign a correct truth label where there exist properties with similar numerical values.

Acknowledgements

This research is supported by a Microsoft Azure for Research grant. Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139).

References

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193.

Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational journalism: A call to arms to database researchers. In *CIDR*, volume 2011, pages 148–151.

Nicholas Diakopoulos. 2016. Accountability in algorithmic decision making. *Communications of the ACM*, 59(2):56–62.

Naeemul Hassan, Bill Adair, James T Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015a. The quest to automate fact-checking. *world*.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015b. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1835–1838. ACM.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3.

Michal Lukasik, Kalina Bontcheva, Trevor Cohn, Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Using gaussian processes for rumour stance classification in social media. *arXiv preprint arXiv:1609.01962*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL System Demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*, pages 1003–1011.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, pages 1589–1599.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. *ACL 2014*, page 18.

Andreas Vlachos and Sebastian Riedel. 2015. Identification and verification of simple claims about statistical properties. In *Proceedings of EMNLP*.

Brett Walenz, You Will Wu, Seokhyun Alex Song, Emre Sonmez, Eric Wu, Kevin Wu, Pankaj K Agarwal, Jun Yang, Naeemul Hassan, Afroza Sultana, et al. 2014. Finding, monitoring, and checking claims computationally based on structured data. In *Computation+ Journalism Symposium*.

ADoCS: Automatic Designer of Conference Schedules

Diego Vallejo[1], **Paulina Morillo**[2], **Cèsar Ferri**[3]
[1]Universidad San Francisco de Quito, Department of Mathematics, Quito, Ecuador
dvallejoh@asig.com.ec
[2]Universidad Politécnica Salesiana, Research Group IDEIAGEOCA, Quito, Ecuador
pmorillo@ups.edu.ec
[3]Universitat Politècnica de València, DSIC, València, Spain
cferri@dsic.upv.es

Abstract

Distributing papers into sessions in scientific conferences is a task consisting in grouping papers with common topics and considering the size restrictions imposed by the conference schedule. This problem can be seen as a semi-supervised clustering of scientific papers based on their features. This paper presents a web tool called ADoCS that solves the problem of configuring conference schedules by an automatic clustering of articles by similarity using a new algorithm considering size constraints.

1 Introduction

Cluster analysis has the objective of dividing data objects into groups, so that objects within the same group are very similar to each other and different from objects in other groups (Tan et al., 2005). Semi-supervised clustering methods try to increase the performance of unsupervised clustering algorithms by using limited amounts of supervision in the form of labelled data or constraints (Basu et al., 2004). These constrains are usually restrictions of size or relations of belonging of objects to the clusters. These membership restrictions have been incorporated into the clustering process by different works (Zhu et al., 2010; Zhang et al., 2014; Ganganath et al., 2014; Grossi et al., 2015) showing that in this context, semi-supervised clustering methods obtain groupings that satisfy the initial restrictions.

On the other hand, in generic form, document clustering should be conceived as the partitioning of a documents collection into several groups according to their content (Hu et al., 2008). A scientific article is a research paper published in specialised journals and conferences. Conferences are usually formed of various sessions of fixed size where the authors present their selected papers. These sessions are usually thematic and are arranged by the conference chair in a manual and tedious work, specially when the number of papers is high. Organising the sessions of a conference can be seen as a problem of document clustering with size constraints.

In this work we present a web application called ADoCS for the automatic configuration of sessions in scientific conferences. The system applies a new semi-supervised clustering algorithm for grouping documents with size constraints. Recently, a similar approach has been described by (Škvorc et al., 2016). In this case the authors also use information from reviews to build the groups, however, this information is not always available.

2 Methodology

In this section we summarise the semi-supervised clustering algorithm of ADoCS system.

The information about the papers in the conference is uploaded to the system by means of a simple csv file. This csv represents a paper per row, and each row must contain, at least, three columns: Title, Keywords and Abstract. For the text preprocessing, NLP techniques and information retrieval techniques are applied to obtain a dissimilarity matrix. We used a classical scheme for data pre-processing in documents: tokenization, stopwords removal and stemming (Jha, 2015).

To structure the dissimilarity matrix of titles and keywords, Jaccard coefficient is applied, since these two elements usually have a small number of tokens. In the case of abstracts a vector model with a cosine similarity index on TF-IDF weighting matrix is used. ADoCS web tool has these default settings, although these parameters can be directly adjusted by the user.

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 41–44

The bag-of-words, or vector model representation derived from the texts, configure a Euclidean space where several distances can be applied in order to estimate similarities between elements. Nevertheless, in some cases, we need to average several criteria and unify them to obtain a single metric that quantifies dissimilarities, and with this, we can derive a distance matrix. This is the case we address here, since for each paper we have three different features: title, abstract and keywords. In these situations we cannot directly apply clustering methods based on centroids, such as K-Means, since there is not a Euclidean space defined for the elements. One way to solve this type of problems is to apply algorithms that, for the clustering process, use only the dissimilarity or distance matrix as input. ADoCS works with a new algorithm called CSCLP (Clustering algorithm with Size Constraints and Linear Programming) (Vallejo, 2016) that only uses as inputs: the size constrains of the sessions and the dissimilarity/distance matrix.

Clustering algorithms obtain better results if a proper selection of initial points method is computed (Fayyad et al., 1998). For this reason, the initial points in our clustering algorithm is chosen using a popular method: Buckshot algorithm (Cutting et al., 1992). In CSCLP, the initial points are used as pairwise constraints (as cannot-link constraints in semi-supervised clustering terminology) for the formation of the clusters, and with binary integer linear programming (BILP) the membership and assignment of the instances to the clusters is determined, satisfying the size constrains of the sessions. In this way, the original clustering problem with size constraints becomes an optimisation problem. Details of the algorithm can be found in (Vallejo, 2016).

3 ADoCS Tool

In this section we include a description of the ADoCS tool. You can find a web version of the tool in the url: `https://ceferra.shinyapps.io/ADoCS`.

On the left part of the web interface we find a panel where we can upload a *csv* file containing information of the papers to be clustered. In the panel, there are several controls where we can configure some features of this *csv* file. Concretely, the separator of fields (comma by default) and how literals are parsed (single quote by default). Additionally, we find three control bars (Title, Keywords and Abstract) with values between 0 and 1 that establish the weights of each of these factors for the computation of distances between papers. By default, the three bars are set to 0.33 to indicate that the three factors will have the same weight in computing the distances. The values of the weights are normalised in such a way that they always sum 1. The user can also configure whether the TF-IDF transformation is applied or not, as well as the metric that is employed to compute the distance between elements. These controls are responsive, i.e., when the user modifies one of the values, the distance matrix is recomputed, and also all the components that depend on this matrix.

Once the file is correctly uploaded in the system, the application enables the function tabs that give access to the functionality of the web system. There are four application tabs:

- **Papers**: This tab contains information about the dataset. We include here the list of papers. For each paper, we show the number, Title, Keywords and Abstract. In order to improve the visualisation, a check box can be employed to show additional information of the papers.

- **Dendrogram**: In this part, a dendrogram generated from the distance matrix is shown. The distance between papers is computed considering the weights selected by the user and the methodology detailed in Section 2.

- **MDS**: In this tab, a Multidimensional Scaling algorithm is employed over the distance matrix to generate a 2D plot about the similarity of papers. Once the clusters are arranged, the membership of the papers to each cluster is denoted by the colour.

- **Wordmap**: This application tab includes a word map representation for showing the most popular terms extracted from the abstracts of the papers in the dataset.

- **Schedule**: In this part, the user can configure the number and size of the sessions and execute the CSCLP algorithm, described in Section 2, to build the groups according the similarity between papers.

The typical use of this tool starts by loading a *csv* file with the information of the papers. After the data is in the system, we can use the first tab *Papers* to know the number of papers in the conference. We can also observe specific information of all the papers (title, keywords, abstract or other information included in the file). The tabs *Dendrogram*, *MDS* and *Wordmap* can be used for exploring the similarities between papers and for knowing the most common terms in the papers of the conference. Finally, we can execute the semi-supervised algorithm to configure the sessions in the *Schedule* tab. In this tab, we can introduce the number of the desired sessions as well as the distribution of the size of the sessions. For instance, if we have 40 papers and we want 5 sessions of 5 papers, and 5 sessions of 3 papers, we introduce "10" in the *Number of Sessions* text box and "5,5,5,5,5,3,3,3,3,3" in *Size of Sessions* text box. When the user push button *compute*, if the sum of the number of papers in the distribution is correct (i.e. is equal to the total number of papers in the conference), the algorithm of section 2 is executed in order to find clusters that satisfy the size restrictions expressed by the session distribution. As a result of the algorithm, a list of the papers with the assigned cluster is shown in the tab. Finally, we can download the assignments as a *csv* file if we push the *Save csv* button.

4 Technology

The ADoCS tool has been implemented totally in R (R Core Team, 2015). This system is a free software language for statistical computing. There is a plethora of different libraries that makes R a powerful environment for developing software related to data analysis in general, and computational linguistics in particular. Specifically, we have employed the following R libraries:

- **tm**: The *tm* package (Meyer et al., 2008) contains a set of functions for text mining in R. In this project we have use the functionalities related to text transformations (stemming, stopwords, TF-IDF, ...) with the English corpora.

- **wordcloud**: The *wordcloud* package (Fellows, 2014) includes functions to show the most common terms in a vocabulary in form of the popular *wordcloud* plot. An example of this plot for the AAAI-2014 conference is included in Figure 1.

Figure 1: Example of the WordMap of the abstracts in the papers of Twenty-Eighth Conference on Artificial Intelligence (AAAI-2014)

- **proxy**: The *proxy* package (Meyer and Buchta, 2016) allows to compute distances between instances. The package contains implementations of popular distances, such as: Euclidean, Manhattan or Minkowski, etc.

- **Rglpk**: The *Rglpk* package (Theussl and Hornik, 2016) represents an R interface to the GNU Linear Programming Kit. GLPK is a popular open source kit for solving linear programming (LP), mixed integer linear programming (MILP) and other related problems with optimisation.

The ADoCS source code and some datasets about conferences to test the tool can be found in `https://github.com/dievalhu/ADoCS`.

The graphical user interface has been developed by means of the Shiny package (Chang et al., 2016). This package constitutes a framework for building graphical applications in R. Shiny is a powerful package for converting basic R scripts into interactive web applications without requiring programming web knowledge. As a result we obtain an interactive web application that can be executed locally using a web browser, or can be uploaded to a shiny application server where the

tool is available for a general use. In this case, we have uploaded the application to the free server *http://www.shinyapps.io/*.

5 Conclusions and Future Work

Arranging papers to create an appropriate conference schedule with sessions containing papers with common topics is a tedious task, specially when the number of papers is high. Machine learning offers techniques that can automatise this task with the help of NLP methods for extracting features from the papers. In this context, organising a conference schedule can be seen as a semi-supervised clustering. In this paper we have presented the ADoCS system, a web application that is able to create a set of clusters according to the similarity of the documents analysed. The groups are formed following the size distribution configured by the user. Although initially the application is focused on grouping conference papers, other related tasks in clustering documents with restrictions could be addressed thanks to the versatility of the interface (different metrics, TF-IDF transformation).

As future work, we are interested in developing conceptual clustering methods to extract topics from the created clusters.

Acknowledgments

This work has been partially supported by the EU (FEDER) and Spanish MINECO grant TIN2015-69175-C4-1-R, LOBASS, by MINECO in Spain (PCIN-2013-037) and by Generalitat Valenciana PROMETEOII/2015/013.

References

Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD conference*, pages 59–68. ACM.

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson, 2016. *shiny: Web Application Framework for R*. R package version 0.14.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/gather: a cluster-based approach to browsing large document collections. *Proceedings of the 15th annual international ACM SIGIR conference*, pages 318–392.

Usama Fayyad, Cory Reina, and Paul S. Bradley. 1998. Initialization of iterative refinement clustering algorithms. *Proceedings of ACMSIGKDD*, pages 194–198.

Ian Fellows, 2014. *wordcloud: Word Clouds*. R package version 2.5.

Nuwan Ganganath, Chi-Tsun Cheng, and Chi. K Tse. 2014. Data clustering with cluster size constraints using a modified k-means algorithm. *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), International Conference IEEE*, pages 158–161.

Valerio Grossi, Anna Monreale, Mirco Nanni, Dino Pedreschi, and Franco Turini. 2015. Clustering formulation using constraint optimization. *Software Engineering and Formal Methods*, pages 93–107.

Guobiao Hu, Shuigeng Zhou, Jihong Guan, and Xiaohua Hu. 2008. Towards effective document clustering: a constrained k-means based approach. *Information Processing & Management*, 44(4):1397–1409.

Monica Jha. 2015. Document clustering using k-medoids. *International Journal on Advanced Computer Theory and Engineering (IJACTE)*, 4(1):2319–2526.

David Meyer and Christian Buchta, 2016. *proxy: Distance and Similarity Measures*. R package version 0.4-16.

David Meyer, Kurt Hornik, and Ingo Feinerer. 2008. Text mining infrastructure in R. *Journal of statistical software*, 25(5):1–54.

R Core Team, 2015. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Tadej Škvorc, Nada Lavrac, and Marko Robnik-Šikonja. 2016. Co-Bidding Graphs for Constrained Paper Clustering. In *5th Symposium on Languages, Applications and Technologies (SLATE'16)*, volume 51, pages 1–13.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2005. *Introduction to data mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Stefan Theussl and Kurt Hornik, 2016. *Rglpk: R/GNU Linear Programming Kit Interface*. R package version 0.6-2.

Diego F. Vallejo. 2016. Clustering de documentos con restricciones de tamaño. Master's thesis, Higher Technical School of Computer Engineering, Polytechnic University of Valencia - UPV.

Shaohong Zhang, Hau-San Wong, and Dongqing Xie. 2014. Semi-supervised clustering with pairwise and size constraints. *International Joint Conference on Neural Networks (IJCNN), IEEE*, pages 2450–2457.

Shunzhi Zhu, Dingding Wang, and Tao. Li. 2010. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889.

A Web Interface for Diachronic Semantic Search in Spanish

Pablo Gamallo, Iván Rodríguez-Torres
Universidade de Santiago de Compostela
Centro Singular de Investigación
en Tecnoloxías da Información (CiTIUS)
15782 Santiago de Compostela, Galiza
pablo.gamallo@usc.es,
ivan.rodriguez.torres@usc.es

Marcos Garcia
Universidade da Coruña
LyS Group
Faculty of Philology
15701 A Coruña, Galiza
marcos.garcia.gonzalez@udc.gal

Abstract

This article describes a semantic system which is based on distributional models obtained from a chronologically structured language resource, namely Google Books Syntactic Ngrams. The models were created using dependency-based contexts and a strategy for reducing the vector space, which consists in selecting the more informative and relevant word contexts. The system allows linguists to analize meaning change of Spanish words in the written language across time.

1 Introduction

Semantic changes of words are as common as other linguistic changes such as morphological or phonological ones. For instance, the word 'artificial' originally meant 'built by the man' (having a positive polarity), but this word has recently changed its meaning if used in contrast with 'natural', aquiring in this context a negative polarity.

Search methods based on frequency (such as Google Trends or Google Books Ngram Viewer search engines) are useful for finding words whose use significantly increases —or decreases— in a specific period of time, but these systems do not not allow language experts to detect semantic changes such as the above mentioned one.

Taking the above into account, this demonstration paper describes a distributional-based system aimed at visualizing semantic changes of words across historical Spanish texts.

The system relies on yearly corpora distributional language models, learned from the Spanish Google Books Ngrams Corpus, so as to obtain word vectors for each year from 1900 to 2009. We compared the pairwise similarity of word vectors for each year and inserted this information in a non structured database. Then, a web interface was designed to provide researchers with a tool, called *Diachronic Explorer*, to search for semantic changes. Diachronic Explorer offers different advanced searching techniques and different visualizations of the results.

Even if there exist similar distributional approaches for languages such as English, German, or French, to the best of our knowledge our proposal is the first work using this technique for Spanish. In addition, we provide the Hispanic community with a useful tool to do linguistic research. The system is open-source[1] and includes a web interface[2] which will be used in the Software Demonstration.

2 The Diachronic Explorer

The Diachronic Explorer relies on a set of distributional semantic models for Spanish language for each year from 1900 to 2009. This semantic resource is stored in a NoSQL database which feeds a web server used for searching and visualizing lexical changes on dozens of thousands of Spanish words along the time.

2.1 Architecture

Figure 1 shows the architecture of Diachronic Explorer. It takes as input a large corpus of dependency-based ngrams to build 110 distributional models, one per year since 1900 to 2009. Then, Cosine similarity is computed for all words in each model in order to generate CSV files consisting of pairs of similar words. Each word is associated with its N (where $N = 20$) most similar words according to the Cosine coefficient. These files are stored in MongoDB to be accessed by web

[1] https://github.com/citiususc/explorador-diacronico

[2] https://tec.citius.usc.es/explorador-diacronico/index.php

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 45–48

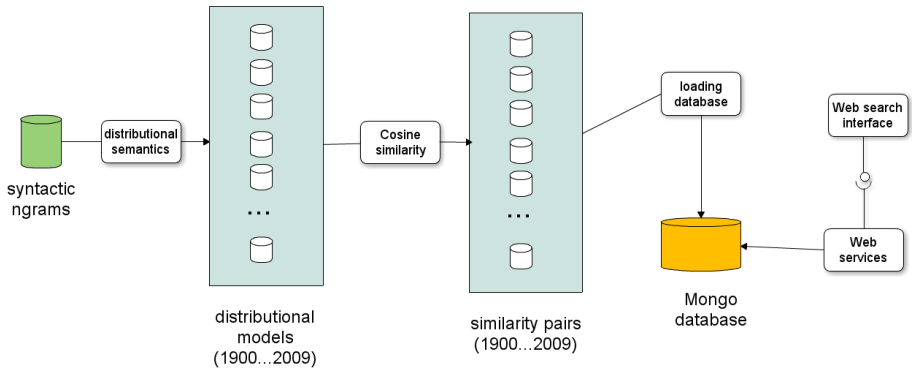


Figure 1: Architecture of Diachronic Explorer

services which generate different types of word searching.

2.2 Yearly Models from Syntactic-Ngrams

To build the distributional semantic models along the time, we made use of the dataset based on the Google Books Ngrams Corpus, which is described in detail in Michel et al. (2011). The whole project contains over 500 billion words (45B in Spanish), being the majority of the content published after 1900. More precisely, we used the Spanish syntactic n-grams from the Version 2012/07/01.[3] Lin et al. (2012) and Goldberg and Orwant (2013) describe the method to extract syntactic ngrams. Each syntactic ngram is accompanied with a corpus-level occurrence count, as well as a time-series of counts over the years. The temporal dimension allows inspection of how the meaning of a word evolves over time by looking at the contexts the word appears in within different time periods.

We transformed the syntactic ngrams into distributional 'word-context' matrices by using a filtering-based strategy that selects for relevant contexts (Gamallo, 2016; Gamallo and Bordag, 2011). A matrix was generated for each year, where each word is represented as a context vector. The final distributional-based resource consists of 110 matrices (one per year from 1900 to 2009). The size of the whole resource is 2,8G, with 25M per matrix in average. Then, the Cosine similarity between word vectors was calculated and, for each word, the 20 most similar ones were selected by year. In total, a data structure with more than 300 million pairs of similar words was generated, giving rise to 110 CSV files (i.e., one file per year).

[3] http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

2.3 Data Storage

We use two ways for storing data. First, we store the distributional models and the similarity pairs in temporal CSV files. These temporal files are generated from offline processes which can be executed periodically as the linguistic input (ngrams) is updated or enlarged with new language sources. Second, the similarity files are imported in a database to make the data access more efficient and easier.

The database system that we chose was MongoDB, which is a NoSQL DB. This type of database system fits well with the task because of two main reasons:

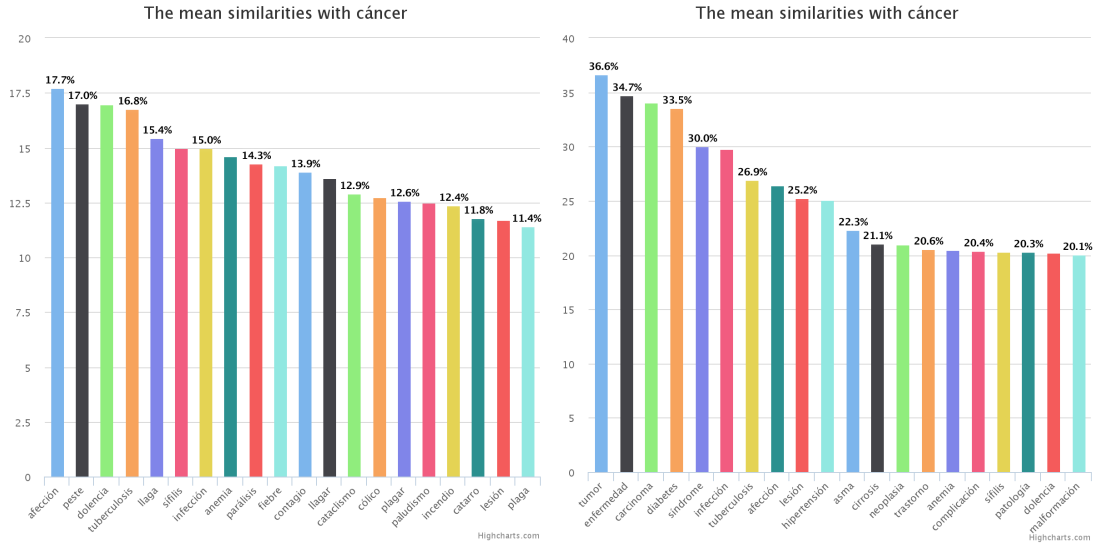
- Our data do not follow a relational model, so we do not need some of the features that make the relational databases powerful, for example, reference keys or table organization.

- NoSQL databases scale really well. Unlike relational databases they implement automaric sharding, which enables us to share and distribute data among servers in the most efficient way possible. So, as the data increase it will be easy to rise up a new instance without any additional engineering.

Among the different types of NoSQL databases, we chose MongoDB because it is document oriented, which fits very well with our data structure.

2.4 Web Interface

The Diachronic Explorer is provided with an web interface that offers different ways of making a diachronic search. Here, we describe just two types of search:



(a) (b)

Figure 2: Similar words to *cáncer* (cancer) using a simple search in 1900 (a) and 2009 (b)

Simple: The user enters a target word and selects for a specific period of time (e.g. from 1920 to 1939), and the system returns the 20 most similar terms (in average) within the selected period of type. The user can select a word cloud image to visualize the output. Figure 2 shows the similar words to *cáncer* ('cancer') returned with a simple search in two different periods: 1900 (a) and 2009 (b). Notice that this word is similar to *peste* and *tuberculosis* at the begining of the 20th century, while nowadays it is more similar to technical words, such as *tumor* or *carcinoma*.

Track record: As in the simple search, the user inserts a word and a time period. However, this new type of search returns the specific similarity scores for each year of the period, instead of the similarity average. In addition, the system allows the user to select any word from the set of all words (and not just the top 20) semantically related to the target one in the searched time period.

3 Conclusions

In this abstract we described a system, *Diachronic Explorer*, that allows linguists to analize meaning change of Spanish words in the written language across time. The system is based on distributional models obtained from a chronologically structured language resource, namely Google Books Syntactic Ngrams. The models were created using dependency-based contexts and a strategy for reducing the vector space, which consists in selecting the more informative and relevant word contexts.

As far as we know, our system is the first attempt to build diachronic distributional models for Spanish language. Besides, it uses NoSQL storage technology to scale easily as new data is processed, and provides an interface enabling useful types of word searching across time. Other similar works for English are reported in different papers (Wijaya and Yeniterzi, 2011; Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kim et al., 2014; Kulkarni et al., 2015).

An interesting direction of research could involve the use of the system infrastructure for holding other types of language variety, namely diatopic variation. Distributional models can be built from text corpora organized, not only with diachronic information, but also with dialectal features. For instance, we could adapt the system to search meaning changes across different diatopic varieties: Spanish from Argentina, Mexico, Spain, Bolivia, and so on. The structure of our system is generic enough to deal with any type of variety, not only that derived from the historical axis.

Acknowledgments

This work has received financial support from a 2016 BBVA Foundation Grant for Researchers and Cultural Creators, TelePares (MINECO,

ref:FFI2014-51978-C2-1-R), *Juan de la Cierva formación* Grant (Ref: FJCI-2014-22853), the Consellería de Cultura, Educación e Ordenación Universitaria (accreditation 2016-2019, ED431G/08) and the European Regional Development Fund (ERDF),

References

Pablo Gamallo and Stefan Bordag. 2011. Is singular value decomposition useful for word simalirity extraction. *Language Resources and Evaluation*, 45(2):95–119.

Pablo Gamallo. 2016. Comparing Explicit and Predictive Distributional Semantic Models Endowed with Syntactic Contexts. *Language Resources and Evaluation*, First online: 13 May 2016. doi:10.1007/s10579-016-9357-4.

Yoav Goldberg and Jon Orwant. 2013. A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (SEM 2013)*, pages 241–247, Atlanta, Georgia.

Kristina Gulordava and Marco Baroni. 2011. A Distributional Similarity Approach to the Detection of Semantic Change in the Google Books Ngram Corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71, Edinburgh, Scotland. ACL.

Adam Jatowt and Kevin Duh. 2014. A Framework for Analyzing Semantic Change of Words Across Time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2014*, pages 229–238, Piscataway, NJ. IEEE Press.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal Analysis of Language through Neural Language Models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65, Baltimore, Maryland. ACL.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically Significant Detection of Linguistic Change. In *Proceedings of the 24th International World Wide Web Conference (WWW 2015)*, pages 625–635, Florence, Italy. ACM.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic Annotations for the Google Books Ngram Corpus. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 169–174. ACL.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 331(6014):176–182.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding Semantic Change of Words over Centuries. In *Proceedings of the 2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the Social Web (DETECT 2011)*, pages 35–40, Glasgow, Scotland. ACM.

Multilingual CALL Framework for Automatic Language Exercise Generation from Free Text

Naiara Perez and **Montse Cuadros**
Vicomtech-IK4 research center
Mikeletegi 57, Donostia-San Sebastián, Spain
{nperez,mcuadros}@vicomtech.org

Abstract

This paper describes a web-based application to design and answer exercises for language learning. It is available in Basque, Spanish, English, and French. Based on open-source Natural Language Processing (NLP) technology such as word embedding models and word sense disambiguation, the system enables users to create automatically and in real time three types of exercises, namely, Fill-in-the-Gaps, Multiple Choice, and Shuffled Sentences questionnaires. These are generated from texts of the users' own choice, so they can train their language skills with content of their particular interest.

1 Introduction

This paper describes a web-based computer-assisted language learning (CALL) framework for automatic generation and evaluation of exercises[1]. The aim of this application is mainly to allow agents in the language learning sector, both teachers and learners alike, to create questionnaires from texts of their own interest with little effort. To do so, the application includes state-of-the-art open source NLP technology, namely part-of-speech tagging, word sense disambiguation, and word embedding models. Its main features are the following:

- **Multilingual.** The platform enables to train Spanish, Basque, English and French skills. Interface messages appear in the chosen language too.
- **Three formats.** Users can design and answer exercises of three types: *a)* Fill-in-the-Gaps (FG): learners must fill in the gaps in a text with the correct words, based on some clues given or just the context provided by the text; *b)* Multiple Choice (MC): learners must choose the correct answers from a set of words given to fill in the gaps in a text; and, *c)* Shuffled Sentences (SS): learners must order a set of given words to formulate grammatical sentences.
- **Highly configurable.** Each exercise format offers a variety of settings that users can control. The input texts from which exercises are built are always given by the users. They also choose the pedagogical target of the exercises based on language-specific part-of-speech (PoS) tags and morphological features. Other settings include the type of clues in FG mode and the amount of distractors in MC. Furthermore, users can select the exercise items themselves or let the system do it automatically.
- **Exportable.** The exercises can be downloaded in Moodle's CLOZE[2] syntax to import them into Moodle quizzes, an extensively exploited platform by teaching institutions all over the world.
- **Evaluation.** The questionnaires designed can be answered in the same application, which prompts the percentage of correct answers upon submission. Correct answers are shown in green and the incorrect ones in red, so the learner can try to guess again.
- **Real-time generation, easy to use.**

[1] The application is at edunlp.vicomtech.org. The web page asks for credentials to log in; the username is "vicomtech", and the password "edunlp". You will also need your own account, which you can obtain by writing an e-mail to the authors of the paper.

[2] https://docs.moodle.org/23/en/Embedded_Answers_(Cloze)_question_type

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 49–52

2 Related Work

There exist countless tools, both web-based and desktop software, that facilitate the creation of teaching material for general purposes, some of the best known being Moodle[3], Hot Potatoes[4], ClassTools.net[5], and ClassMarker[6]. However, the focus of such tools is on enabling users to adapt the pedagogical content they are interested in, whichever it is, to certain exercise formats (i.e., quizzes, open clozes, crosswords, drag-and-drops, and so on). That is, these tools do not offer support for assessing the contents on their pedagogical suitability nor other exercise-dependent tasks, such as building clues for quizzes or distractors for multiple-choice exercises.

In the domain of language learning in particular, exercise authoring very often implies a lot of word list curation, searching for texts that contain certain linguistic patterns or expressions, retrieving definitions, and similar tasks. The availability of resources for language teachers that simplify these processes, such as on-line dictionaries or teaching-oriented lexical databases (e.g., English Profile[7]), depends on the target language. To the best of this paper's authors' knowledge, there does not exist at the moment an exercise generation and evaluation framework specific to learn Basque, Spanish, English, and French, that not only automatizes formatting the content given by the user for several exercises but also incorporates natural language processing (NLP) techniques to ease the authoring process. Volodina et al. (2014) describe a similar framework named Lärka. Lärka designs multiple-choice exercises in Swedish for linguistics or Swedish learners. The questions are based on controlled corpora, that is, the users cannot choose the texts they will be working on.

3 Workflow description

All the exercise formats mentioned share a common building process. Users must choose a language, provide a text –in that language–, and choose a pedagogical target from the options given. Pedagogical targets are based on PoS tags and morphological features. Different possible targets have been implemented for each language, depending on the languages' characteristics and the richness of the parsing models available. For instance, exercises in Spanish can target the subjunctive conjugation or the definite/indefinite articles. In English, one can target, for example, past and present participle tenses. Once the initial configuration has been set, the workflow continues as follows:

3.1 Extraction of candidate items

The text given is segmented, tokenized and tagged with IXA pipes (Agerri et al., 2014), using the latest models provided with the tools. The tokens with a PoS tag or morphological feature selected by the user as pedagogical target are chosen as candidate items. In the case of the SS format, item candidates are sentences containing tokens with the relevant PoS tags or morphological features. If the text does not contain any candidate item, the application alerts the user that it is not possible to build an exercise with the configuration given.

3.2 Final item selection

The system has two ways of getting the final items from the candidates identified: the user can choose whether to select them or let the application do it randomly. In the latter case, the user can set an upper bound to the amount of items generated. The system never yields two contiguous items, since it would increase substantially the difficulty in answering them.

3.3 Exercise generation

Once the final items have been chosen, the actual questionnaire must be designed. This depends on the format chosen by the user and the specific settings available to that format:

Fill-in-the-Gaps (FG). The system substitutes the items chosen with gaps that have to be filled in with the appropriate words. Users can choose to show, for all the languages, at least three types of clues to help learners do the exercise: the lemmas of the correct words, their definition, or a word bank of all the correct words (and how many times they occur). For Spanish and English, the system is also capable of prompting the morphological features of the words to be guessed, in addition to the lemma. This feature is interesting to train on singulars and plurals, grammatical gender, verbal tenses, and so on. Moreover, depending on the pedagogical target chosen, the system automatically disables certain types of clues. It would not

[3] https://moodle.org/
[4] https://hotpot.uvic.ca/
[5] https://www.classtools.net/
[6] https://www.classmarker.com/
[7] http://www.englishprofile.org/

make much sense, for instance, to give the lemmas of the correct words if the pedagogical target were prepositions, given that prepositions cannot be lemmatized. The user can also choose to not give any help.

To generate clues based on lemmas and/or morphologial features, the system turns to the linguistic annotation given by the IXA pipes during candidate extraction. The annotation contains all the information necessary for each token in the text.

Retrieving definitions requires additional processing, since a word's definition depends on the context it appears in. That is, choosing the correct definition of a word in the text provided by the user translates to disambiguating the word. The application relies on Babelfy (Moro et al., 2014) and BabelNet (Navigli and Ponzetto, 2010) APIs in order to do so. It passes the whole text as the context and asks Babelfy to assign a single sense to the words chosen as targets of the exercise. Then, it retrieves from BabelNet the definitions associated to those senses. Babelfy is not always able to assign a sense to a word; when this happens, the application returns the lemma of the word as its clue.

Multiple Choice (MC). Again, the exercise consists of gaps in the text to be filled with the correct words. In this case, the learner is given a set of words from which to choose an answer. This set of words contains the right answer and some incorrect words called "distractors". When this exercise format is chosen, the system automatically generates as many distractors as specified by the user. This is achieved by consulting, for each correct answer, a word embedding model and retrieving the most similar words. The models are word2vec (Mikolov et al., 2013b; Mikolov et al., 2013a) trained on Leipzig University's corpora[8] with the library Gensim for Python[9]. Thus, distractors tend to be words that appear often in contexts similar to the right answer, but not semantically or grammatically correct. Distractors are then transformed to the same case as the correct word and finally shuffled for their visualization.

Shuffled Sentences (SS). This exercise consists in ordering a set of words given to formulate a grammatical sentence. In this case, the system substitutes the sentences chosen by gaps and shows the sentence shuffled as a lead.

[8] http://corpora.uni-leipzig.de/

[9] http://radimrehurek.com/gensim/

3.4 Evaluation

The user can answer the questionnaire it has designed and get it assessed by the system. For all the exercise formats, the evaluation consist in comparing the correct answers with the input received from the user. The answer is right only when it is the same to the correct answer.

4 The demonstrator

The application has a clean interface and is easy to follow. An exercise can be designed, answered and evaluated visiting less than four pages:

The home page. In the home page, the user chooses a language –Spanish, Basque, English or French–, and an exercise format –FG, MC, or SS. It leads to the exercise configuration page of the exercise chosen.

The configuration page. All the configuration pages share a common structure. A text field occupies the top of the page. This is where the user introduces the text they want to work with. Below the text field are the configuration options, presented as radio-button lists. All the exercise formats require that users choose a pedagogical target. Then come the format-specific settings, the only section of the page that varies.

For an FG exercise, two properties must be set: the type of clue and how the clues must be visualized. They can be shown below the gapped text with a reference to the gap they belong to, or as description boxes of the gaps (i.e., "tooltips").

In a MC exercise, users must choose the amount of distractors they want the system to create.

For the SS mode, users can set an upper limit to the sentences that will be selected as candidates.

Finally, the user can choose whether to let the system select the items of the exercise or choose them themselves among the candidates that the system generates. In the former case, the system asks the user how many items it should create at most, and takes the user directly to the "Answer and evaluate" page. In the latter case, the user is taken to the "Choose the items" page.

If the text does not contain any token that meets the pedagogical target, the system asks the user to provide a different text or change the configuration set. That is, the application allows for users to know with a single click whether the texts they choose are suitable for the pedagogical objective they have set, without them having had previously read the text thoroughly.

Choose the items. In this page users choose the items it wants to create among the words that meet the pedagogical target they chose. The interface shows the whole text with all the available candidates selected. They can be toggled simply by clicking on them. There are also buttons to select all and remove all the candidates. Once users are satisfied with their selection, they are led to the "Answer and evaluate" page.

Answer and evaluate. This is the page where the final questionnaire is displayed and can be filled. The appearance varies a little depending on the format chosen. FG exercises consist of the text given and gaps where the correct words should be written. If the tooltip clues option has been enabled, clues appear in description boxes when hovering over the gaps; otherwise, they appear listed below the text. Word bank clues appear boxed on to the right of the text. In MC exercises, the choices are radio-button groups listed to the right of the text. As for SS exercises, page-wide gaps replace the sentences chosen, and the shuffled words are given above the gaps.

At the bottom of this page there is a link to download the exercise in Moodle CLOZE syntax. The file that is downloaded can be imported to Moodle in order to generate a quiz.

Users can fill in the exercise they designed and submit them to get the percentage of correct answers. Furthermore, the answers are colored in green or red, depending on whether they are correct or not, respectively. This way learners can try to answer again.

5 Conclusions

We have described a web-based framework for language learning exercise generation. It is available for Spanish, Basque, English, and French. Users can design three types of exercises to train diverse skills in these languages. The framework is highly configurable and lets the user choose whether they want to select the exercise items or have the system do it.

As future work, the system should be improved in various ways. To begin with, the actual system delegates to the user the task of making the exercises more or less difficult by choosing the items themselves. The application will be endowed with technology that allows the users to create automatically exercises which vary in difficulty starting from the same text.

Another aspect that can be improved is the fact that exercise items are created from unigrams. It would be very interesting that the application were capable of generating multi-word candidates. This would be useful to revise collocations or phrasal verbs, for instance. In this same regard, the applicability of the system would increase if it based item candidate generation not only on PoS tags or morphological features, but on other criteria as well like the semantics of the text.

There is also room for improvement in the configurability of the application. Users should be allowed to control two key features: definition clues in FG exercises and MC distractors. Currently, the application imposes the definitions and distractors it generates, instead of presenting them as options for the users to choose.

Finally, we plan to implement more formats that add to the available three.

Acknowledgements

This work has been funded by the Spanish Government (project TIN2015-65308-C5-1-R).

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. IXA pipeline: Efficient and Ready to Use Multilingual NLP tools. In *LREC*, volume 2014, pages 3823–3828.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeff Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

Andrea Moro, Francesco Cecconi, and Roberto Navigli. 2014. Multilingual word sense disambiguation and entity linking for everybody. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*, pages 25–28. CEUR-WS. org.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *ACL*, pages 216–225.

Elena Volodina, Ildikó Pilán, Lars Borin, and Therese Lindström Tiedemann. 2014. A flexible language learning platform based on language resources and web services. In *LREC*, pages 3973–3978.

Audience Segmentation in Social Media

Verena Henrich, Alexander Lang
IBM Germany Research and Development GmbH
Böblingen, Germany
verena.henrich@de.ibm.com, alexlang@de.ibm.com

Abstract

Understanding the social media audience is becoming increasingly important for social media analysis. This paper presents an approach that detects various audience attributes, including author location, demographics, behavior and interests. It works both for a variety of social media sources and for multiple languages. The approach has been implemented within *IBM Watson Analytics for Social Media*™ and creates author profiles for more than 300 different analysis domains every day.

1 Understanding the Social Media Audience – Why Bother?

The *social media audience* shows *Who* is talking about a company's products and services in social media. This is increasingly important for various teams within an organization:

Marketing: *Does our latest social media campaign resonate more with men or with women? What are social media sites where actual users of our products congregate? What other interests do authors have that talk about our products, so we can create co-selling opportunities?*

Sales: *Which people are disgruntled with our products or services and consider churning? Can we find social media authors interested in buying our type of product, so we can engage with them?*

Product management and product research: *Which product features are important specifically for women, or parents? What aspects do actual users of our product highlight—and how does this compare to the competition's product?*

Besides commercial scenarios, social media is becoming relevant for the social and political sciences to understand opinions and attitudes towards various topics. Audience insights are key to put these opinions into the right context.

2 Audience Segmentation with IBM Watson Analytics for Social Media

IBM Watson Analytics for Social Media™ (WASM) is a cloud-based social media analysis tool for line-of-business users from public relations, marketing or product management[1]. The tool is *domain-independent*: users configure *topics* of interest to analyze, e.g., products, brands, services or politics. Based on the user's topics, WASM retrieves all relevant content from *a variety of social media sources* (Twitter, Facebook, blogs, forums, reviews, video comments and news) across *multiple languages* (currently Arabic, English, French, German, Italian, Portuguese and Spanish) and applies various natural language processing steps:

- Dynamic topic modeling to spot ambiguous user topics, and suggest variants.
- Aspect-oriented sentiment analysis.
- Detection of spam and advertisements.
- Audience segmentation, including author location, demographics, behavior, interests, and account types.

While the system demo will show all steps, this paper focuses on *audience segmentation*. Audience segmentation relies on: (i) The author's *name* and *nick name*. (ii) The author's *biography*: a short, optional self-description of the author (see Figure 1), which often includes the author's location. (iii) The author's *content*: social media posts (Tweets, blog posts, forum entries, reviews) that contain topics configured by a WASM user.

[1] https://watson.analytics.ibmcloud.com

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 53–56

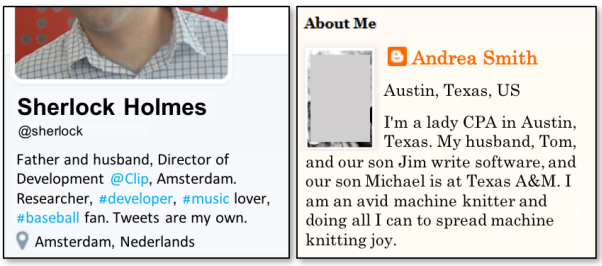


Figure 1: Example author biographies.

Our segmentation approach only relies on the 'on-topic' content of an author, not on all of the author's posts—for two reasons: Firstly, many social media sources such as forums do not allow retrieving content based on an author name. Secondly, social media content providers charge per document. It would not be economical to download all content for all authors who talk about a certain topic. We found that the combination of the author's content with the author's name(s) and biography significantly enhances both precision and recall of audience segmentation.

2.1 Author Location

WASM detects the *permanent* location of an author, i.e., the city or region where the author lives, by matching the information found in their biography against large, curated lists of city, region and country names. We use population information (preferring the location with higher population) to disambiguate city names in the absence of any region or country information, e.g., an author with the location *Stuttgart* is assigned to Stuttgart, Germany, not Stuttgart, AR, USA.

2.2 Author Demographics

WASM identifies three demographic categories:

Marital status: When the author is married, the value is *true*, otherwise*unknown*. For the classification we identify keywords (encoded in dictionaries) and patterns in the author's biography and content. Our dictionaries and patterns match, for example, authors describing themselves as *married* or *husband*, or authors that use phrases such as *my spouse loves running* or *at my wife's birthday*. Thus, WASM tags the authors in Figure 1 as married.

Parental status: When the author has children, the value is *true*, otherwise *unknown*. Similarly to the marital status classification, keywords and patterns are matched in the authors' biographies and contents. Example keywords are *father* or *mom*; example patterns are *my kids* or *our daughter*. WASM tags both authors in Figure 1 as having children.

Gender: Possible values are *male*, *female* and *unknown*. The classification of gender relies on a similar keyword and pattern matching as described previously. In addition, it relies on matching the author name against a built-in database of 150,000 first names that span a variety of cultures. For ambiguous first names such as *Andrea* (which identifies females in Germany and males in Italy), we take both the language of the author content and the detected author location into account to pick the appropriate gender. WASM classifies the first author in Figure 1 as male, the second author as female.

2.3 Author Behavior

WASM identifies three types of author behavior: **Users** own a certain product or use a particular service. They identify themselves through phrases such as *my new PhoneXY*, *I've got a PhoneXY* or *I use ServiceXY*. **Prospective users** are interested in buying a product or service. Example phrases are *I'm looking for a PhoneXY* or *I really need a PhoneXY*. **Churners** have either quit using a product or service, or have a high risk of leaving. They are identified by phrases such as *Bye Bye PhoneXY* or *I sold my PhoneXY*.

Author behavior is classified by matching keywords and patterns in the authors biographies and content—similarly to the demographic analysis described above. It allows to understand: *What do actual customers of a certain product or service talk about? What are the key factors why customers churn?*

Figure 2 shows an example analysis where the topics of interest were three retailers: a discounter, an organic market and a regular supermarket.[2] The top of Figure 2 summarizes what is relevant for authors that WASM identified as *user*s, i.e., customers of a specific retailer. The bottom part shows two social media posts from *user*ss.

2.4 Author Interests

WASM defines a three-tiered taxonomy of interests, which is inspired by the IAB Tech Lab Content Taxonomy (Interactive Advertising Bureau, 2016). The first tier represents eight top-level interest categories such as *art and entertainment*

[2]The retailers' names are anonymized.

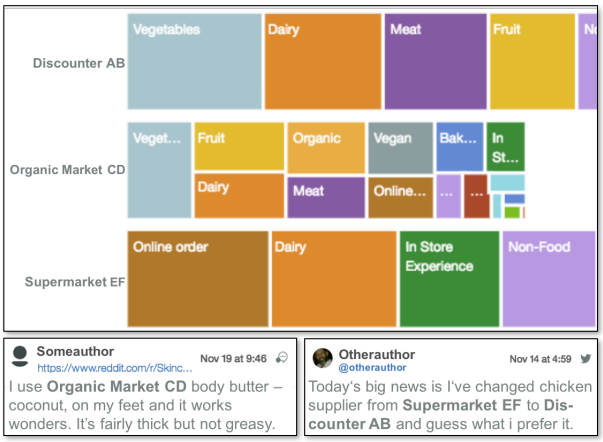


Figure 2: What matters to users of supermarkets.

or *sports*. The second tier comprises about 60 more specific categories. These include *music* and *movies* under *art and entertainment* and *ball sports* and *martial arts* under *sports*. The third level represents fine-grained interests, e.g., *tennis* and *soccer* below *ball sports*.

The fine-grained interests on the third level are identified in author biographies and content with the help of dictionaries and patterns. From the biography of the first author in Figure 1, we infer that he is interested in music and baseball (*music lover* and *baseball fan*). For the second author we infer an interest in machine knitting (*I am an avid machine knitter*). Note that a simple occurrence of a certain interest, e.g. a sport type, is usually not enough: it has to be qualified in the author content by matching specific patterns. A match in a biography, however, typically qualifies as a bona fide interest—excluding, e.g., negation structures.

Figure 3 visualizes the connections between the three retailers mentioned in the previous chapter and coarse-grained interest categories. This reveals that authors who write about *Organic Market CD* are uniquely interested in animals and that *Discounter AB* does not attract authors who are interested in sports. These insights allow targeted advertisements or co-selling opportunities.

Figure 3: Author interests related to Retailers

2.5 Account Type Classification

WASM classifies social media accounts into *organizational* or *personal*. This helps customers to understand who is really driving the social media conversations around their brands and products: is it themselves (through official accounts), a set of news sites, or is it an 'organic' conversation with lots of personal accounts involved?

Organizational accounts include companies, NGOs, newspapers, universities, and fan sites. Personal accounts are non-organizational accounts from individual authors. Account types are distinguished by cues from the authors' names and biographies. For example, company-indicating expressions such as *Inc* or *Corp* and patterns like *Official Twitter account of <some brand>* indicate that the account represents an organization, whereas a phrase like *Views are my own* points to an actual person. The biographies in Figure 1 contain many hints that both accounts are personal, e.g., agent nouns (*director*, *developer*, *lover*) and personal and possessive pronouns (*I*, *my*, *our*).

3 Implementation

3.1 Text Analysis Stack

The WASM author segmentation components are created by using the *Annotation Query Language* (AQL; Chiticariu et al., 2010). AQL is an SQL-style programming language for extracting structured data from text. Its underlying relational logic supports a clear definition of rules, dictionaries, regular expressions, and text patterns. AQL optimizes the rule execution to achieve higher analysis throughput.

The implemented rules harness linguistic preprocessing steps, which consist of tokenization, part-of-speech tagging and chunking (Abney, 1992; Manning and Schütze, 1999)—the latter also expressed in AQL. Rules in AQL support the combination of tokens, parts of speech, chunks, string constants, dictionaries, and regular expressions. Here is a simplified pattern for identifying whether an author has children:

```
create view Parent as
extract pattern
  <FP.match> <A.match>{0,1} /child|sons?|kids?/
from FirstPersonPossessive FP, Adjectives A;
```

It combines a dictionary of first person possessive pronouns with an optional adjective (identified by its part of speech) and a regular expression matching child words. The pattern matches

phrases such as *my son* and *our lovely kids*.

3.2 System Architecture

We wanted to provide users analysis results as quickly as possible. Hence, we created a *data streaming* architecture that retrieves documents from social media, and analyzes them on the fly, while the retrieval is still in progress. Moreover, we wanted to run all analyses for all users on a single, *multi-tenant* system to keep operating costs low. The data stream that our text analysis components see contains social media content for different users. Hence, we built components that can switch between different analysis processing configurations with minimal overhead.

The text analysis components run as Scala or Java modules within Apache Spark™. We exploit Spark's Streaming API for our data streaming requirements. The data between the components flows through Apache Kafka™. The combination of Spark and Kafka allows for processing scale-out, and resilience against component failures.

The multi-tenant text analysis processing is separated into user-specific and language-specific analysis steps. We optimized the user-specific analysis steps (such as detecting products and brands a particular user cares about) to have virtually no switching overhead. The language-specific steps (e.g., sentiment detection or author segmentation) are invariant across customers. To achieve the required processing throughput, we launch one language-specific rule set per processing node, and analyze multiple documents in parallel with this language-specific rule set.

The analysis pipeline runs on a cluster of 10 virtual machines, each with 16 cores and 32G RAM. Our customers run 300+ analyses per day, analyzing between 50,000 to 5,000,000 documents each.

4 Evaluation

Our customers are willing to accept smaller segment sizes (i.e., lower recall) as long as the precision of the segment identification is high. This design goal of our analysis components is reflected in the evaluation results for author behavior and account types on English social media documents that we present here.[3]

The retail dataset mentioned in previous chapters consists of 50,354 documents by 41,209 authors from all social media sources. WASM classifies 1,695 authors as *users*—without any additional effort by the customer. Compare that with the task to run a survey in the retailer's stores (and its competition) to get similar insights. We manually annotated author behavior for 500 documents from distinct authors. The evaluated precision is 90.0% at a recall of 58.3%.

The evaluation of account types is based on a random sample of 50,124 Tweets, which comprises 43,193 distinct authors. 36,682 of the authors provide biographies. We use this as the "upper recall bound" for our biography-based approach. Our system assigns an account type for 18,657 authors, which corresponds to a recall of 50.9%. It classifies 16,981 as *personal* and 1,676 as *organizational* accounts. We manually annotated 500 of the classified accounts, which results in a precision of 97.4%.

5 Conclusion and Future Work

This paper presented real-life use cases that require understanding audience segments in social media. We described how IBM Watson Analytics for Social Media™ segments social media authors according to their location, demographics, behavior, interests and account types. Our approach shows that the author biography in particular is a rich source of segment information. Furthermore, we show how author segments can be created at a large scale by combining natural language processing and the latest developments in data analytics platforms. In future work, we plan to extend this approach to additional author segments as well as additional languages.

[3] An evaluation of each audience feature for each supported language is beyond the scope of this paper.

References

Steven P. Abney, 1992. *Parsing By Chunks*, pages 257–278. Springer Netherlands, Dordrecht.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. SystemT: An Algebraic Approach to Declarative Information Extraction. In *Proceedings of ACL*, pages 128–137.

Interactive Advertising Bureau. 2016. IAB Tech Lab Content Taxonomy. Online, accessed: 2016-12-23, https://www.iab.com/guidelines/iab-quality-assurance-guidelines-qag-taxonomy/.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

The arText prototype: An automatic system for writing specialized texts

Iria da Cunha
Universidad Nacional de Educación a Distancia (UNED)
Senda del Rey 7, 28040
Madrid, Spain
iriad@flog.uned.es

M. Amor Montané
Universitat Pompeu Fabra
Roc Boronat 138, 08018
Barcelona, Spain
amor.montane@upf.edu

Luis Hysa
Universitat Pompeu Fabra
Roc Boronat 122, 08018
Barcelona, Spain
luis.hysa@gmail.com

Abstract

This article describes an automatic system for writing specialized texts in Spanish. The arText prototype is a free online text editor that includes different types of linguistic information. It is designed for a variety of end users and domains, including specialists and university students working in the fields of medicine and tourism, and laypersons writing to the public administration. ArText provides guidance on how to structure a text, prompts users to include all necessary contents in each section, and detects lexical and discourse problems in the text.

1 Introduction

In the field of Natural Language Processing (NLP), various types of linguistic information including phonological, morphological, lexical, syntactic, semantic and discourse-related features can be used to develop applications. To date, tools for writing texts have often been designed for general subject areas and included information on orthographic, grammatical and/or lexical aspects of the writing process. NLP researchers have tended not to study systems for structuring and writing specialized texts, although a few researchers have bucked this trend: Kinnunen et al. (2012) developed a system to identify and correct writing problems in English in several domains; Aluisio et al. (2001)'s system helps non-native speakers write scientific publications in English; the Writing Pal (Dai et al., 2011) and Estilector[1] systems help improve academic writing in English and Spanish, respectively; and LanguageTool[2] is an open source proofreading program for non-specialized texts in several languages. To our knowledge, none of the systems that are currently available have considered the specific characteristics of textual genres in specialized domains, such as medicine, tourism and the public administration.

Writing specialized texts is more challenging than writing general texts (Cabré, 1999). Textual, lexical and discourse features are an essential component of textual genres, such as medical research papers, travel blog posts, or claims submitted to the public administration. Against this backdrop, this article aims to present a prototype for an automatic system that provides assistance in writing specialized texts in Spanish. The arText system includes textual, lexical and discourse-related information, and is useful for different end users. It provides guidance on how to structure a text, prompts users to include all necessary contents in each section, and detects lexical and discourse problems in the text.

Da Cunha et al. (in press) determined the most frequent textual genres that pose the greatest writing challenges for three groups: specialists and university students in medicine and tourism, and laypersons writing to the public administration. ArText was designed to help these users write the 15 textual genres included in Table 1.

Section 2 describes the characteristics of the system and its modules. Section 3 explains how the system was evaluated, while Section 4 presents conclusions and future lines of research.

2 Description of the System

ArText is a free online text editor that anyone can use, with no registration required. The system was developed in a LINUX environment using an Apache server and a MySQL database. A variety of resources were utilized in the back end (BASH, PERL, and PHP, with a Laravel Frame-

[1] http://www.estilector.com/index.php.
[2] https://www.languagetool.org/.

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 57–60

Medicine	Research article Review article Medical history Abstract Bachelor's thesis
Tourism	Informative article Travel blog post Report Rules and regulations Business plan
Public Administration	Allegation Cover letter Letter of complaint Claim Application

Table 1: Specialized fields and textual genres included in arText.

work) and front end (HTML, CSS, JAVASCRIPT, with AJAX and JQUERY); Google Analytics is integrated into the site to measure traffic.

Documents can be exported in four formats: PDF, TXT, HTML and ARTEXT. Previously saved documents can be uploaded using the ARTEXT format, and the website includes a detailed user manual and a contact section for comments, questions and suggestions.

ArText can be accessed at `http://sistema-artext.com/`,[3] and has been optimized for the Google Chrome browser. To use arText, click on "Start using arText" and pick one of the 15 textual genres mentioned above. This brings you to the text editor, where you can start writing using the text editor and the three modules integrated into arText: Structure, Contents and Phraseology; Format and Spellchecking; and Lexical and Discourse-based Recommendations.

2.1 Module 1. Structure, Contents and Phraseology

The left-hand column helps users structure and draft documents. Its interactive template includes typical sections, contents and phraseology for each textual genre. This information was extracted from da Cunha and Montané (2016), a corpus-based analysis following van Dijk (1989)'s textual approach. Specifically, users can insert:

- Typical document sections
- Typical contents found in each section
- Phraseology related to each of these contents

The text editor displays the sections which typically appear in a given textual genre. For example, the template for a "claim" to be submitted to the public administration includes the following sections:

- Header
- Addressee
- Introductory clause
- Supporting details
- Request
- Closing

A drop-down menu in the left-hand column provides sample texts for each section, including section titles, where appropriate. For example, the "Supporting details" section includes two different contents:

- Grounds for the claim
- Attachments

When users click on a specific content, arText displays a list of sample phrases that can be incorporated into the final text. For example, "Attachments" includes the following phrases:

- Attached please find [document name].[4]
- The following supporting documents are attached: [list of documents].

Users can click on a stock phrase to include it in the text.

2.2 Module 2. Format and Spellchecking

The toolbar at the top of the screen includes an open source spellchecker (WebSpellChecker Ltd.) and various formatting options, e.g. to change font or font size; insert bullet points, images, tables and links; cut, copy and paste; print; and search. Since online storage is not provided, the user's manual includes instructions for uploading an image to Google Drive and inserting it into a document produced using arText.

2.3 Module 3. Lexical and Discourse-based Recommendations

By clicking on the review button in the right-hand column, users can see a series of lexical and discourse-related recommendations for improving their texts. These recommendations are derived from da Cunha and Montané (2016), which is based on Cabré (1999)'s Communicative Theory of Terminology and Mann and Thompson (1988)'s Rhetorical Structure Theory. The module includes a series of algorithms based on linguistic rules and two NLP tools: the Freeling shallow parser (At-

[3] A demo is available at `https://canal.uned.es/mmobj/index/id/54433`

[4] Users are instructed to fill in the information indicated in square brackets (e.g. names, dates and numbers).

serias et al., 2006) and the DiSeg discourse segmenter (da Cunha et al., 2012).

This module includes 11 main recommendations, all of which are displayed in the right-hand column, when appropriate. A subset of these recommendations is assigned to each textual genre, and all recommendations are adapted to the linguistic characteristics of each genre (da Cunha and Montané, 2016). Recommendations cover the following 11 topics:

1. Spelling out acronyms
2. Using acronyms systematically
3. Providing definitions
4. Using the passive voice
5. Using the 1st person systematically
6. Using subjectivity indicators
7. Repeating words
8. Using long sentences
9. Segmenting long sentences
10. Considering alternative discourse markers
11. Varying discourse markers

By clicking on a given recommendation, users can see a more detailed explanation and suggestions. In some cases, arText also highlights phrases or content in the text editor. For example, one lexical recommendation, "Spelling out acronyms," highlights acronyms that are not spelled out when they first appear in the text (i.e. arText would highlight "COPD" if "chronic obstructive pulmonary disease" did not appear next to this acronym the first time the term was used). Some recommendations also actively engage users in the revision process. For example, the recommendation "Repeating words" shows a list of repeated words; when users click on a word in the right-hand column, all occurrences of this word in the text are highlighted. This recommendation is not displayed for highly specialized textual genres (e.g. research articles and abstracts), since lexical variation is usually avoided in these types of texts (Cabré, 1999).

Some recommendations focus on the discourse level. For example, "Segmenting long sentences" highlights one or more long sentences; users can click to see suggestions for splitting them into shorter sentences. In this case, arText proposes these discourse segments in the right-hand column. The number of words used to determine long sentences differs for each textual genre, following da Cunha and Montané (2016).

Another discourse level recommendation refers to "Varying discourse markers." In this case, arText displays a list of discourse markers repeated in the text. When users click on one of these markers, all of its occurrences are highlighted, and a list of alternative discourse markers used to express the same relationship (e.g. Cause, Restatement, Contrast and Condition, etc.) is displayed in the right-hand column. For instance, for the discourse marker "that is," used to express Restatement, arText suggests the alternatives "in other words," "that is to say," "i.e." and "to put it another way."

3 Evaluation

Real and *ad hoc* texts were used to test arText's algorithms and linguistic rules and improve the system. Subsequently, the prototype was launched and data-driven and user-driven evaluations were conducted.

The data-driven evaluation was based on a test corpus with 24 texts corresponding to one textual genre from each domain; the corpus comprised eight medical abstracts, eight tourism-related informative articles and eight applications to the public administration. The linguistic characteristics of these texts were manually annotated, and the manual annotation and arText results were compared. Precision and recall were measured for a series of recommendations; the results are presented in Table 2.[5]

Recommendation ID	Precision	Recall
1	0.76	0.68
2	0.75	0.94
3	x	x
4	1	0.94
5: sing. verbal forms	0.87	0.97
5: pl. verbal forms	1	0.99
6	-	-
9	0.74	0.87
10	1	1

Table 2: Data-driven results.

Recommendation 7 did not apply in the medical subcorpus; in the tourism and public administration subcorpora, 91.67% and 94.70% of detected words, respectively, were repeated in the text. For Recommendation 8, 100% of highlighted sentences in the medicine and tourism subcorpora were long sentences according to the thresholds for abstracts and informative articles; no long sentences appeared in the public administration

[5] In light of the degree of specialization, Recommendation 4 did not apply for any of the textual genres included in the test corpus. No cases for which Recommendation 6 applies were found in the test corpus.

subcorpus, so this recommendation could not be tested for this genre. No cases of Recommendation 11 were found in the medical and administration subcorpora; in the tourism corpus, 100% of detected discourse markers were repeated in the text, and adequate alternatives were proposed.

The user-driven evaluation aimed to determine how useful arText is. A survey designed using Google Forms focused on accessibility, the usefulness of the three modules and general issues. Three doctors, three tourism professionals, and 25 laypersons completed the survey; all laypersons were between 30-50 years old and had both higher education experience and internet skills. In general, respondents found arText to be user-friendly and useful; 100% of them would recommend the system to other people. Respondents found the section on structure to be the most useful module, while the approach to uploading images was considered the system's greatest weakness.

4 Conclusions and Future Work

This paper describes a prototype of an automatic system to assist users in writing specialized texts. The online arText editor helps users draft texts for 15 textual genres in three specialized domains: medicine, tourism and the public administration. It lays out the structure for each section of the document, suggests appropriate contents and stock phrases for each section, and detects typical linguistic errors. This innovative system is the first tool that considers lexical, textual and discourse features for specific textual genres. Moreover, the arText project is based on the idea that academic research can be shared with and used constructively by the general public.

In the future, the results of the data-driven evaluation will be utilized to improve arText's algorithms. A second user-driven evaluation will include a broader population (e.g. students). Finally, arText may be adapted to other textual genres, specialized domains and languages.

Acknowledgments

This article is part of the "Automatic system to help in writing specialized texts in domains relevant to Spanish society" research project, funded by "2015 BBVA Foundation Grants for Researchers and Cultural Creators". It was also supported by a Ramón y Cajal contract (RYC-2014-16935). We would like to thank Josh Goldsmith for the proofreading of the text, and the evaluation survey respondents and the research groups ACTUALing and IULATERM for their support.

References

Sandra M. Aluísio, Iris Barcelos, Jandir Sampaio, and Osvaldo N. Oliveira Jr. 2001. How to learn the many unwritten "rules of the game" of the academic discourse: A hybrid approach based on critiques and cases to support scientific writing. *Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2001*.

Jordi Atserias, Bernardino Casas, Elisabet Comelles, Meritxell González, Lluís Padró, and Muntsa Padró. 2006. Freeling 1.3. syntactic and semantic services in an open-source nlp library. *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 48–55.

M. Teresa Cabré. 1999. *La Terminología. Representación y comunicación*. Institut for Applied Linguistics, Barcelona.

Iria da Cunha and M. Amor Montané. 2016. Un análisis lingüístico de los géneros textuales del ámbito de la administración con repercusión en la vida de los ciudadanos. *Proceedings of the XV Symposium of the Ibero-American Terminology Network (RITerm 2016)*, pages 61–62.

Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, Marina Lloberes, and Irene Castellón. 2012. Diseg 1.0: The first system for spanish discourse segmentation. *Expert Systems with Applications*, 39(2):1671–1678.

Iria da Cunha, M. Amor Montané, and Alba Coll. in press. Detección de géneros textuales que presentan dificultades de redacción: un estudio en los ámbitos de la administración, la medicina y el turismo. *Proceedings of the 34th International Conference of the Spanish Association of Applied Linguistics*.

Jianmin Dai, Roxanne B. Raine, Rod Roscoe, Zhiqiang Cai, and Danielle S. McNamara. 2011. The writing-pal tutoring system: Development and design. *Journal of Engineering and Computer Innovations*, 2(1):1–11.

Tomi Kinnunen, Henri Leisma, Monika Machunik, Tuomo Kakkonen, and Jean-Luc Lebrun. 2012. Swan scientific writing assistant a tool for helping scholars to write reader-friendly manuscripts. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–24.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–288.

Teun A. van Dijk. 1989. *La ciencia del texto*. Paidós, Barcelona.

QCRI Live Speech Translation System

Fahim Dalvi, Yifan Zhang, Sameer Khurana, Nadir Durrani, Hassan Sajjad
Ahmed Abdelali, Hamdy Mubarak, Ahmed Ali, Stephan Vogel
Qatar Computing Research Institute
Hamad bin Khalifa University, Doha, Qatar
{faimaduddin, yzhang}@qf.org.qa

Abstract

We present QCRI's Arabic-to-English speech translation system. It features modern web technologies to capture live audio, and broadcasts Arabic transcriptions and English translations simultaneously. Our Kaldi-based ASR system uses the Time Delay Neural Network architecture, while our Machine Translation (MT) system uses both phrase-based and neural frameworks. Although our neural MT system is slower than the phrase-based system, it produces significantly better translations and is memory efficient.[1]

1 Introduction

We present our Arabic-to-English SLT system consisting of three modules, the Web application, Kaldi-based Speech Recognition culminated with Phrase-based/Neural MT system. It is trained and optimized for the translation of live talks and lectures into English. We used a Time Delayed Neural Network (TDNN) for our speech recognition system, which has a word error rate of 23%. For our machine translation system, we deployed both the traditional phrase-based Moses and the emerging Neural MT system. The trade-off between efficiency and accuracy (BLEU) barred us from picking only one final system. While the phrase-based system was much faster (translating 24 tokens/second versus 9.5 tokens/second), it was also roughly 5 BLEU points worse (28.6 versus 33.6) compared to our Neural MT system. We therefore leave it up to the user to decide whether they care more about translation quality or speed. The *real-time* factor for the entire pipeline is 1.18 using Phrase-based MT and 1.26 using Neural MT.

[1]The demo is available at https://st.qcri.org/demos/livetranslation.

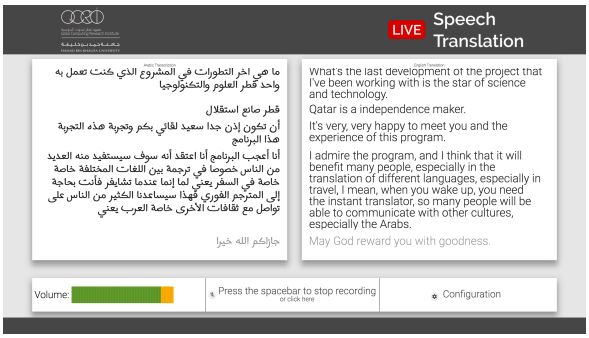


Figure 1: Speech translation system in action. The Arabic transcriptions and English translations are shown in *real-time* as they are decoded.

Our system is also robust to common English code-switching, frequent acronyms, as well as dialectal speech. Both the Arabic transcriptions and the English translations are presented as results to the viewers. The system is built upon modern web technologies, allowing it to run on any browser that has implemented these technologies. Figure 1 presents a screen shot of the interface.

2 System Architecture

The QCRI live speech translation system is primarily composed of three fairly independent modules: the web application, speech recognition, and machine translation. Figure 2 shows the complete work-flow of the system. It mainly involves the following steps: 1) Send audio from a broadcast instance to the ASR server; 2) Receive transcription from the ASR server; 3) Send transcription to MT server; 4) Receive translation from the MT server; 5) Sync results with backend system and 6) Multiple watch instances sync results from the backend. Steps 1-5 are constantly repeated as new audio is received by the system through the broadcast page. Step 6 is also periodically repeated to get the latest results on the watch page. Both the speech recognition and machine translation mod-

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 61–64

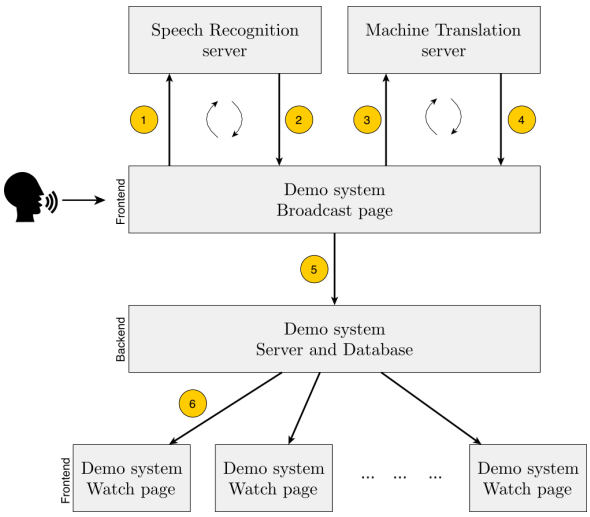


Figure 2: Demo system overview

ules have a standard API that can be used to send and receive information. The *Web Application* connects with the API and runs independently of the system used for transcription and translation.

2.1 Web application

The web application has two major components; the frontend and the backend. The frontend is created using the React Javascript framework to handle the dynamic User Interface (UI) changes such as transcription and translation updates. The backend is built using NodeJS and MongoDB to handle sessions, data associated with these sessions and authentication. The frontend presents the user with three pages; the landing page, the watch page and the broadcast page. The landing page allows the user to either create a new session or work with an existing one. The watch page regularly syncs with the backend to get the latest partial or final transcriptions and translations. The broadcast page is meant for the primary speaker. This page is responsible for recording the audio data and collecting the transcriptions and translations from the ASR and MT systems respectively. Both partial transcriptions and translations are also presented to the speaker as they are being decoded. To avoid very frequent and abrupt changes, the rate of update of partial translations was configured based on a `MIN_NEW_WORDS` parameter, which defines the minimum number of new words required in the partial transcription to trigger the translation service. Both the partial and the final results are also synced to the backend as they are made available, so that the viewers on the watch page can experience the live translation.

2.2 Speech transcription

We use the Speech-to-text transcription system that was built as part of QCRI's submission to the 2016 Arabic Multi-Dialect Broadcast Media Recognition (MGB) Challenge. Key features of the transcription system are given below:

Data: The training data consisted of 1200 hours of transcribed broadcast speech data collected from Aljazeera news channel. In addition we had 10 hours of development data (Ali et al., 2016). We used data augmentation techniques such as Speed and Volume perturbation which increased the size of the training data to three times the original size (Ko et al., 2015).

Speech Lexicon: We used a Grapheme based lexicon (Killer and Schultz, 2003) of size 900k. The lexicon is constructed using the words that occur more than twice in the training transcripts.

Speech Features: Features used to train all the acoustic models are 40 dimensional hi-resolution Mel Frequency Cepstral Coefficients (MFCC_hires), extracted for each speech frame, concatenated with 100 dimensional i-Vectors per speaker to facilitate speaker adaptation (Saon et al., 2013).

Acoustic Models: We experimented with three acoustic models; Time Delayed Neural Networks (TDNNs) (Peddinti et al., 2015), Long Short-Term Memory Recurrent Neural Networks (LSTM) and Bi-directional LSTM (Sak et al., 2014). Performance of the BLSTM acoustic model in terms of *Word Error Rate* is better than the TDNN, but TDNN has a much better *real-time* factor while decoding. Hence, for the purpose of the speech translation system, we use the TDNN acoustic model. The TDNN model consists of 5 hidden layers, each layer containing 1024 hidden units and is trained using Lattice Free Maximum Mutual Information (LF-MMI) modeling framework in Kaldi (Povey et al., 2016). Word Error Rate comparison of different acoustic models can be seen in Table 1. For further details, see Khurana and Ali (2016).

Language Model: We built a Kneser Ney smoothed trigram language model. The vocab size is restricted to the 100k most frequent words to improve the decoding speed and the *real-time factor* of the system. The choice of using a trigram model instead of an RNN as in our offline systems was essential in keeping the decoding speed at a reasonable value.

Decoder Parameters: Beam size for the de-

Model	%WER
TDNN	23.0
LSTM	20.9
BLSTM	19.3

Table 1: Recognition results for the LF-MMI trained recognition systems. LM used for decoding is tri-gram. Data augmentation is used before training

coder was tuned to give the best real-time factor with a reasonable drop in accuracy. The final value was selected to be 9.0.

2.3 Machine translation

The MT component is served by an API that connects to several translation systems and allows the user to seamlessly switch between them. We had four systems to choose from for our demo, two of which were Phrase-based systems, and the two were Neural MT systems trained using Nematus (Sennrich et al., 2016).

PB-Best: This is a competition-grade phrase-based system, also used for our participation at the IWSLT'16 campaign (Durrani et al., 2016). It was trained using all the freely available Arabic-English data with state-of-the-art features such as a large language model, lexical reordering, OSM (Durrani et al., 2011) and NNJM features (Devlin et al., 2014).

PB-Pruned: The PB-best system is not suitable for real time translation and has high memory requirements. To increase the efficiency, we dropped the OSM and NNJM features, heavily pruned the language model and used MML-filtering to select a subset of training data. The resulting system was trained on 1.2 M sentences, 10 times less the original data.

NMT-GPU: This is our best system[2] that we submitted to the IWSLT'16 campaign (Durrani et al., 2016). The advantage of Neural models is that their size does not scale linearly with the data, and hence we were able to train using all available data without sacrificing translation speed. This model runs on the GPU.

NMT-CPU: This is the same model as 3, but runs on the CPU. We use the AmuNMT (Junczys-Dowmunt et al., 2016) decoder to use our neural models on the CPU. Because of computation constraints, we reduced the beam size from 12 to 5 with a minimal loss of 0.1 BLEU points.

The primary factors in our final decision were 3-fold; overall quality, translation time and computational constraints. The translation time has to be small for a live translation system. The performance of the four systems on the official IWSLT test-sets is shown in Figure 3.

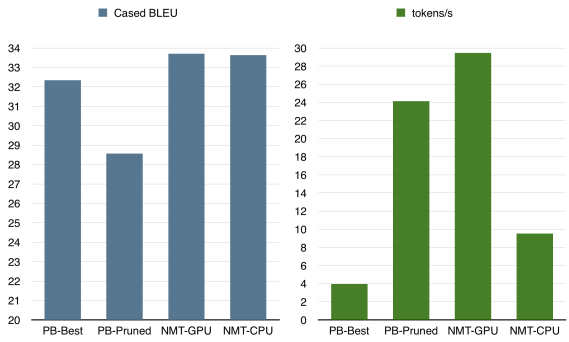


Figure 3: Performance and Translation speed of various MT systems

We also computed the translation speed of each of the systems.[3] The results shown in Figure 3 depict the significant time gain we achieved using the pruned phrase based system. However, with a 5 BLEU point difference in translation quality, we decided to compromise and use the slower **NMT-CPU** in our final demo. We also allow the user to switch to the phrase-based system, if translation speed is more important. We did not use **NMT-GPU** since it is very costly to put into production with its requirement for a dedicated GPU card.

Finally, we added a customized dictionary and translated unknown words by transliterating them in a post-decoding step (Durrani et al., 2014).

2.4 Combining Speech recognition and Machine translation

To evaluate our complete pipeline, we prepared three in-house test sets. The first set was collected from an in-house promotional video, while the other two sets were collected in a quiet office environment.

[2] without performing ensembling

[3] **PB-Pruned** and **NMT-CPU** were run using a single CPU thread on our standard demo machine using a `Intel(R) Xeon(R) E5-2660 @ 2.20GHz` processor. **PB-Best** was run on another machine using a `Intel(R) Xeon(R) E5-2650 @ 2.00GHz` processor due to memory constraints. Finally, **NMT-GPU** was run using an `NVidia GeForce GTX TITAN X` GPU card.

We analyzed the real time performance of the entire pipeline, include the lag induced by translation after the transcription is complete. With an average *real-time factor* of 1.1 for the speech recognition, our system keeps up with normal speech without any significant lag. The distribution of the real time factor speech recognition and translation for the in-house test sets is shown in Figure 4.

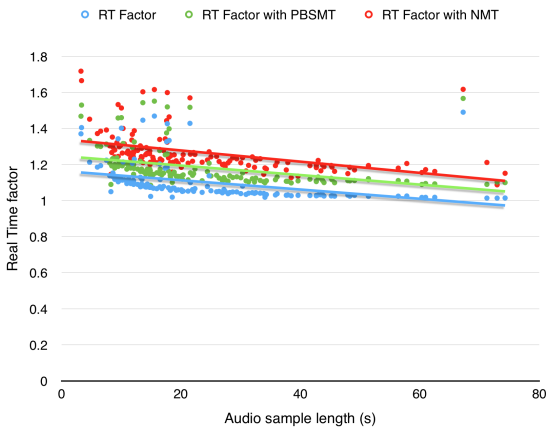


Figure 4: Real-time analysis for all audio segments in our in-house test sets

3 Conclusion

This paper presents QCRI live speech translation system for real world settings such as lectures and talks. Currently, the system works very well for Arabic including frequent dialectal words, and also supports code-switching for most common acronyms and English words. Our future aim is to improve the system in several ways; by having a tighter integration between the speech recognition and translation components, incorporating more dialectal speech recognition and translation, and by improving punctuation recovery of the speech recognition system which will help machine translation to produce better translation quality.

References

Ahmed Ali, Peter Bell, James Glass, Yacine Messaoui, Hamdy Mubarak, Steve Renals, and Yifan Zhang. 2016. The mgb-2 challenge: Arabic multi-dialect broadcast media recognition. In *SLT*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11)*, Portland, OR, USA.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, volume 14, pages 148–153.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and Stephan Vogel. 2016. QCRI machine translation systems for IWSLT 16. In *Proceedings of the 15th International Workshop on Spoken Language Translation*, IWSLT '16, Seattle, WA, USA.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. *CoRR*, abs/1610.01108.

Sameer Khurana and Ahmed Ali. 2016. QCRI advanced transcription system (QATS) for the arabic multi-dialect broadcast media recognition: MGB-2 challenge. In *Spoken Language Technology Workshop (SLT) 2016 IEEE*.

Mirjam Killer and Tanja Schultz. 2003. Grapheme based speech recognition. In *INTERSPEECH*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH*.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, pages 3214–3218.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. 2013. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, pages 55–59.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany, August. Association for Computational Linguistics.

Nematus: a Toolkit for Neural Machine Translation

Rico Sennrich[†] **Orhan Firat**[⋆] **Kyunghyun Cho**[‡] **Alexandra Birch**[†]
Barry Haddow[†] **Julian Hitschler**[¶] **Marcin Junczys-Dowmunt**[†] **Samuel Läubli**[§]
Antonio Valerio Miceli Barone[†] **Jozef Mokry**[†] **Maria Nădejde**[†]
[†]University of Edinburgh [⋆]Middle East Technical University
[‡]New York University [¶]Heidelberg University [§]University of Zurich

Abstract

We present Nematus, a toolkit for Neural Machine Translation. The toolkit prioritizes high translation accuracy, usability, and extensibility. Nematus has been used to build top-performing submissions to shared translation tasks at WMT and IWSLT, and has been used to train systems for production environments.

1 Introduction

Neural Machine Translation (NMT) (Bahdanau et al., 2015; Sutskever et al., 2014) has recently established itself as a new state-of-the art in machine translation. We present Nematus[1], a new toolkit for **Ne**ural **Ma**chine **T**ranslation.

Nematus has its roots in the dl4mt-tutorial.[2] We found the codebase of the tutorial to be compact, simple and easy to extend, while also producing high translation quality. These characteristics make it a good starting point for research in NMT. Nematus has been extended to include new functionality based on recent research, and has been used to build top-performing systems to last year's shared translation tasks at WMT (Sennrich et al., 2016) and IWSLT (Junczys-Dowmunt and Birch, 2016).

Nematus is implemented in Python, and based on the Theano framework (Theano Development Team, 2016). It implements an attentional encoder–decoder architecture similar to Bahdanau et al. (2015). Our neural network architecture differs in some aspect from theirs, and we will discuss differences in more detail. We will also describe additional functionality, aimed to enhance usability and performance, which has been implemented in Nematus.

[1] available at `https://github.com/rsennrich/nematus`

[2] `https://github.com/nyu-dl/dl4mt-tutorial`

2 Neural Network Architecture

Nematus implements an attentional encoder–decoder architecture similar to the one described by Bahdanau et al. (2015), but with several implementation differences. The main differences are as follows:

- We initialize the decoder hidden state with the mean of the source annotation, rather than the annotation at the last position of the encoder backward RNN.
- We implement a novel conditional GRU with attention.
- In the decoder, we use a feedforward hidden layer with $\tanh$ non-linearity rather than a maxout before the softmax layer.
- In both encoder and decoder word embedding layers, we do not use additional biases.
- Compared to *Look*, *Generate*, *Update* decoder phases in Bahdanau et al. (2015), we implement *Look*, *Update*, *Generate* which drastically simplifies the decoder implementation (see Table 1).
- Optionally, we perform recurrent Bayesian dropout (Gal, 2015).
- Instead of a single word embedding at each source position, our input representations allows multiple features (or "factors") at each time step, with the final embedding being the concatenation of the embeddings of each feature (Sennrich and Haddow, 2016).
- We allow tying of embedding matrices (Press and Wolf, 2017; Inan et al., 2016).

We will here describe some differences in more detail:

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 65–68

Table 1: Decoder phase differences

RNNSearch (Bahdanau et al., 2015)		Nematus (DL4MT)	
Phase	Output - Input	Phase	Output - Input
Look	$\mathbf{c}_j \leftarrow \mathbf{s}_{j-1}, \mathrm{C}$	Look	$\mathbf{c}_j \leftarrow \mathbf{s}_{j-1}, y_{j-1}, \mathrm{C}$
Generate	$y_j \leftarrow \mathbf{s}_{j-1}, y_{j-1}, \mathbf{c}_j$	Update	$\mathbf{s}_j \leftarrow \mathbf{s}_{j-1}, y_{j-1}, \mathbf{c}_j$
Update	$\mathbf{s}_j \leftarrow \mathbf{s}_{j-1}, y_j, \mathbf{c}_j$	Generate	$y_j \leftarrow \mathbf{s}_j, y_{j-1}, \mathbf{c}_j$

Given a source sequence $(x_1, \ldots, x_{T_x})$ of length T_x and a target sequence $(y_1, \ldots, y_{T_y})$ of length T_y, let $\mathbf{h}_i$ be the annotation of the source symbol at position i, obtained by concatenating the forward and backward encoder RNN hidden states, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$, and $\mathbf{s}_j$ be the decoder hidden state at position j.

decoder initialization Bahdanau et al. (2015) initialize the decoder hidden state $\mathbf{s}$ with the last backward encoder state.

$$\mathbf{s}_0 = \tanh\left(\mathbf{W}_{init}\overleftarrow{\mathbf{h}}_1\right)$$

with $\mathbf{W}_{init}$ as trained parameters.[3] We use the average annotation instead:

$$\mathbf{s}_0 = \tanh\left(\mathbf{W}_{init}\frac{\sum_{i=1}^{T_x}\mathbf{h}_i}{T_x}\right)$$

conditional GRU with attention Nematus implements a novel conditional GRU with attention, cGRU_{att}. A cGRU_{att} uses its previous hidden state $\mathbf{s}_{j-1}$, the whole set of source annotations $\mathrm{C} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{T_x}\}$ and the previously decoded symbol y_{j-1} in order to update its hidden state $\mathbf{s}_j$, which is further used to decode symbol y_j at position j,

$$\mathbf{s}_j = \text{cGRU}_{\text{att}}\left(\mathbf{s}_{j-1}, y_{j-1}, \mathrm{C}\right)$$

Our conditional GRU layer with attention mechanism, cGRU_{att}, consists of three components: two GRU state transition blocks and an attention mechanism ATT in between. The first transition block, GRU_1, combines the previous decoded symbol y_{j-1} and previous hidden state $\mathbf{s}_{j-1}$ in order to generate an intermediate representation $\mathbf{s}'_j$ with the following formulations:

$$\begin{aligned}
\mathbf{s}'_j &= \text{GRU}_1\left(y_{j-1}, \mathbf{s}_{j-1}\right) = (1 - \mathbf{z}'_j) \odot \underline{\mathbf{s}}'_j + \mathbf{z}'_j \odot \mathbf{s}_{j-1}, \\
\underline{\mathbf{s}}'_j &= \tanh\left(\mathbf{W}'\mathbf{E}[y_{j-1}] + \mathbf{r}'_j \odot (\mathbf{U}'\mathbf{s}_{j-1})\right), \\
\mathbf{r}'_j &= \sigma\left(\mathbf{W}'_r\mathbf{E}[y_{j-1}] + \mathbf{U}'_r\mathbf{s}_{j-1}\right), \\
\mathbf{z}'_j &= \sigma\left(\mathbf{W}'_z\mathbf{E}[y_{j-1}] + \mathbf{U}'_z\mathbf{s}_{j-1}\right),
\end{aligned}$$

where $\mathbf{E}$ is the target word embedding matrix, $\underline{\mathbf{s}}'_j$ is the proposal intermediate representation, $\mathbf{r}'_j$ and $\mathbf{z}'_j$ being the reset and update gate activations. In this formulation, $\mathbf{W}'$, $\mathbf{U}'$, $\mathbf{W}'_r$, $\mathbf{U}'_r$, $\mathbf{W}'_z$, $\mathbf{U}'_z$ are trained model parameters; σ is the logistic sigmoid activation function.

The attention mechanism, ATT, inputs the entire context set C along with intermediate hidden state $\mathbf{s}'_j$ in order to compute the context vector $\mathbf{c}_j$ as follows:

$$\begin{aligned}
\mathbf{c}_j &= \text{ATT}\left(\mathrm{C}, \mathbf{s}'_j\right) = \sum_i^{T_x}\alpha_{ij}\mathbf{h}_i, \\
\alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{Tx}\exp(e_{kj})}, \\
e_{ij} &= \mathbf{v}_a^\intercal \tanh\left(\mathbf{U}_a\mathbf{s}'_j + \mathbf{W}_a\mathbf{h}_i\right),
\end{aligned}$$

where α_{ij} is the normalized alignment weight between source symbol at position i and target symbol at position j and $\mathbf{v}_a, \mathbf{U}_a, \mathbf{W}_a$ are the trained model parameters.

Finally, the second transition block, GRU_2, generates $\mathbf{s}_j$, the hidden state of the cGRU_{att}, by looking at intermediate representation $\mathbf{s}'_j$ and context vector $\mathbf{c}_j$ with the following formulations:

$$\begin{aligned}
\mathbf{s}_j &= \text{GRU}_2\left(\mathbf{s}'_j, \mathbf{c}_j\right) = (1 - \mathbf{z}_j) \odot \underline{\mathbf{s}}_j + \mathbf{z}_j \odot \mathbf{s}'_j, \\
\underline{\mathbf{s}}_j &= \tanh\left(\mathbf{W}\mathbf{c}_j + \mathbf{r}_j \odot (\mathbf{U}\mathbf{s}'_j)\right), \\
\mathbf{r}_j &= \sigma\left(\mathbf{W}_r\mathbf{c}_j + \mathbf{U}_r\mathbf{s}'_j\right), \\
\mathbf{z}_j &= \sigma\left(\mathbf{W}_z\mathbf{c}_j + \mathbf{U}_z\mathbf{s}'_j\right),
\end{aligned}$$

similarly, $\underline{\mathbf{s}}_j$ being the proposal hidden state, $\mathbf{r}_j$ and $\mathbf{z}_j$ being the reset and update gate activations with the trained model parameters $\mathbf{W}, \mathbf{U}, \mathbf{W}_r, \mathbf{U}_r, \mathbf{W}_z, \mathbf{U}_z$.

Note that the two GRU blocks are not individually recurrent, recurrence only occurs at the level of the whole cGRU layer. This way of combining RNN blocks is similar to what is referred in the literature as *deep transition* RNNs (Pascanu et al., 2014; Zilly et al., 2016) as opposed to the more common *stacked* RNNs (Schmidhuber, 1992; El Hihi and Bengio, 1995; Graves, 2013).

deep output Given $\mathbf{s}_j$, y_{j-1}, and $\mathbf{c}_j$, the output probability $p(y_j|\mathbf{s}_j, y_{j-1}, \mathbf{c}_j)$ is computed by a softmax activation, using an intermediate representation $\mathbf{t}_j$.

$$\begin{aligned}
p(y_j|\mathbf{s}_j, y_{j-1}, \mathbf{c}_j) &= \text{softmax}\left(\mathbf{t}_j\mathbf{W}_o\right) \\
\mathbf{t}_j &= \tanh\left(\mathbf{s}_j\mathbf{W}_{t1} + \mathbf{E}[y_{j-1}]\mathbf{W}_{t2} + \mathbf{c}_j\mathbf{W}_{t3}\right)
\end{aligned}$$

$\mathbf{W}_{t1}, \mathbf{W}_{t2}, \mathbf{W}_{t3}, \mathbf{W}_o$ are the trained model parameters.

[3] All the biases are omitted for simplicity.

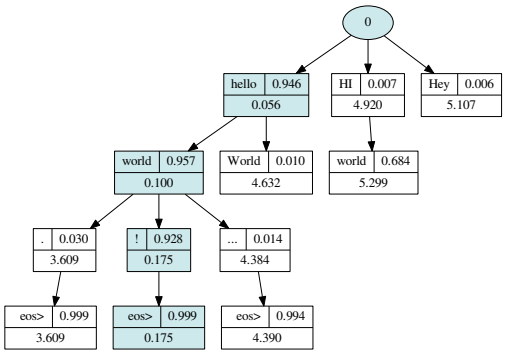


Figure 1: Search graph visualisation for DE→EN translation of "Hallo Welt!" with beam size 3.

3 Training Algorithms

By default, the training objective in Nematus is cross-entropy minimization on a parallel training corpus. Training is performed via stochastic gradient descent, or one of its variants with adaptive learning rate (Adadelta (Zeiler, 2012), RmsProp (Tieleman and Hinton, 2012), Adam (Kingma and Ba, 2014)).

Additionally, Nematus supports minimum risk training (MRT) (Shen et al., 2016) to optimize towards an arbitrary, sentence-level loss function. Various MT metrics are supported as loss function, including smoothed sentence-level BLEU (Chen and Cherry, 2014), METEOR (Denkowski and Lavie, 2011), BEER (Stanojevic and Sima'an, 2014), and any interpolation of implemented metrics.

To stabilize training, Nematus supports early stopping based on cross entropy, or an arbitrary loss function defined by the user.

4 Usability Features

In addition to the main algorithms to train and decode with an NMT model, Nematus includes features aimed towards facilitating experimentation with the models, and their visualisation. Various model parameters are configurable via a command-line interface, and we provide extensive documentation of options, and sample set-ups for training systems.

Nematus provides support for applying single models, as well as using multiple models in an ensemble – the latter is possible even if the model architectures differ, as long as the output vocabulary is the same. At each time step, the probability distribution of the ensemble is the geometric average of the individual models' probability distributions. The toolkit includes scripts for beam search decoding, parallel corpus scoring and n-best-list rescoring.

Nematus includes utilities to visualise the attention weights for a given sentence pair, and to visualise the beam search graph. An example of the latter is shown in Figure 1. Our demonstration will cover how to train a model using the command-line interface, and showing various functionalities of Nematus, including decoding and visualisation, with pre-trained models.[4]

5 Conclusion

We have presented Nematus, a toolkit for Neural Machine Translation. We have described implementation differences to the architecture by Bahdanau et al. (2015); due to the empirically strong performance of Nematus, we consider these to be of wider interest.

We hope that researchers will find Nematus an accessible and well documented toolkit to support their research. The toolkit is by no means limited to research, and has been used to train MT systems that are currently in production (WIPO, 2016).

Nematus is available under a permissive BSD license.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 645452 (QT21), 644333 (TraMOOC), 644402 (HimL) and 688139 (SUMMA).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Boxing Chen and Colin Cherry. 2014. A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA.

[4] Pre-trained models for 8 translation directions are available at `http://statmt.org/rsennrich/wmt16_systems/`

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical Recurrent Neural Networks for Long-Term Dependencies. In *Nips*, volume 409.

Yarin Gal. 2015. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *ArXiv e-prints*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *CoRR*, abs/1611.01462.

Marcin Junczys-Dowmunt and Alexandra Birch. 2016. The University of Edinburgh's systems submission to the MT task at IWSLT. In *The International Workshop on Spoken Language Translation (IWSLT)*, Seattle, USA.

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

Razvan Pascanu, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to Construct Deep Recurrent Neural Networks. In *International Conference on Learning Representations 2014 (Conference Track)*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation, Volume 1: Research Papers*, pages 83–91, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 368–373, Berlin, Germany.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany.

Milos Stanojevic and Khalil Sima'an. 2014. BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal, Quebec, Canada.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5 - rmsprop.

WIPO. 2016. WIPO Develops Cutting-Edge Translation Tool For Patent Documents, Oct. `http://www.wipo.int/pressroom/en/articles/2016/article\_0014.html`.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

A tool for extracting sense-disambiguated example sentences through user feedback

Beto Boullosa[†] and **Richard Eckart de Castilho**[†]
and **Alexander Geyken**[‡] and **Lothar Lemnitzer**[‡]
and **Iryna Gurevych**[†]

[†]**Ubiquitous Knowledge Processing Lab**
Department of Computer Science
Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

[‡]**Berlin-Brandenburg Academy of Sciences**
http://www.bbaw.de

Abstract

This paper describes an application system aimed to help lexicographers in the extraction of example sentences for a given headword based on its different senses. The tool uses classification and clustering methods and incorporates user feedback to refine its results.

1 Introduction

Language is subject to constant evolution and change. Hence, lexicographers are always several steps behind the current state of language in discovering new words, new senses of existing words, cataloging them, and illustrating them using good example sentences. To facilitate this work, lexicographers increasingly rely on automatic approaches that allow sifting efficiently through the ever growing body of digitally available text, something that has brought important gains, including time saving and better utilizing limited financial and personal resources.

Among the tasks that benefit from the increasing automation in lexicography is the automatic extraction of suitable corpus-based example sentences for the headwords in the dictionary. Our paper describes an innovative system that handles this task by incorporating user feedback to a computer-driven process of sentence extraction based on a combination of unsupervised and supervised machine learning, in contrast to current approaches that do not include user feedback. Our tool allows querying sentences containing a specific lemma, clustering these sentences by topical similarity to initialize a sense classifier, and interactively refining the sense assignments, continually updating the classifier in the background.

In the next section, we contextualize the task of example extraction; section 3 describes our system; section 4 is devoted to evaluation; section 5 summarizes the conclusions and future work.

2 Extraction of Dictionary Examples

Example sentences can help understanding the nuances of the usage of a certain term, specially in the presence of polysemy. This has become rather important in the last decades, with the shift that has occurred, in the field of dictionary making, from a content-centered to a user-centered perspective (Lew, 2015). With the popularization of online dictionaries, space-saving considerations have lost the importance once held, making it easier to add example sentences to a given headword.

Didakowski et al. (2012) argue that a system for example extraction should ideally act "like a lexicographer", i.e., it should fully understand the examples themselves, something arguably beyond the scope of current NLP technology. Instead, operational criteria must be used to define "good" examples, as seen also in the work of Kilgarriff et al. (2008), criteria like presence of typical collocations of the target word, characteristics of the sentence itself, and guaranteeing that all senses of the target word are represented by the extracted example sentences.

Several methods to automate the task have been developed, the most popular being GDEX ("Good Dictionary EXamples") (Kilgarriff et al., 2008). GDEX is a rule based software tool that suggests "good" corpus examples to the lexicographer according to predefined criteria, including sentence length, word frequency and the presence/absence of pronouns or named entities. The goal of GDEX is to reduce the number of corpus examples to be inspected by extracting only the n-"best" examples, the default being 100 sentences. It has been used and adapted for languages other than English.

Didakowski et al. (2012) presented an extrac-

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 69–72

tor of good examples based on a hand-written, rule-based grammar, determining the quality of a sentence according to its grammatical structure combined with some simpler features as used by GDEX. None of those works, however, focused on differentiating example sentences according to the word senses or the target words.

Cook et al. (2013) used Word Sense Induction (WSI) to identify novel senses for words in a dictionary. They utilized hierarchical LDA (Latent Dirichlet Allocation) (Teh et al., 2006), a variation of the original LDA (Blei et al., 2003) topic model, to identify novel word senses, later combining this approach with GDEX, to allow extracting good example sentences according to word senses (Cook et al., 2014). However, they obtained "encouraging rather than conclusive" results, specially due to limitations of the LDA approach in linking identified topics with word senses.

Our work explores and develops a similar approach of using topic modeling and WSI to cluster sentences according to the senses of a target word, but we take a step further, using the initial clusters as seed for a series of interactive classification steps. The training data for each classification step are sentences whose confidence scores calculated by the system exceed a threshold, and sentences manually labeled by the user. The process leads to a user-driven refinement in the labeling process.

3 System description

The computer-assisted, interactive sense disambiguation process supported by our system involves: 1) import sentences into a searchable index; 2) retrieve sentences containing a specific word (lemma); 3) cluster selected sentences, providing a starting point for interactive classification; 4) train a multi-class classifier from the initial clusters - that trains on the most representative sentences for each cluster and is then used to label the rest of the sentences; a sentence is "representative" if the confidence score calculated by the system exceeds a configurable threshold; 5) refine the classifier by interactively correcting sense assignments for selected sentences.

The system supports multiple users working in so called **projects**, which define, among other things, the list of stopwords available for clustering and classification and the location where the sentences should be retrieved from.

A project contains **jobs**, corresponding to tasks performed over a certain headword (actually defined by its lemma and POS tag). Tasks include searching for initial sentences, clustering and classification. Users can work on many jobs in parallel and isolated, which allows calculating inter-rater agreement on the sense disambiguation task.

As for the technologies, the tool was developed using Java Wicket and relying on Solr[1], Mallet [2], DKPro Core (Eckart de Castilho and Gurevych, 2014), Tomcat and MySQL. The next subsections describe the system in more detail.

3.1 Searching

After starting a job, the user goes to the Search page to look for sentences containing the desired lemma. They are shown in a KWIC (Keyword in Context) table with their ids. When satisfied with the results, the user selects the stopwords to use in the next steps and goes to the Clustering phase.

3.2 Clustering

In the clustering page, the selected sentences are automatically divided into clusters corresponding to topics that ideally relate to the senses of the target word. The user manually configures how many clusters the topic modeling generates and control the hyper-parameters to fine-tune the process. We currently use Mallet's LDA implementation to topic modeling (McCallum, 2002).

The clustering page lists the selected sentences according to the generated clusters. Each cluster is shown in its own column, with a word cloud on the top, containing the main words related to it, which helps the user to assess cluster's quality and meaning. The word sizes correspond to their LDA-calculated weights. The user can change hyper-parameters and regenerate clusters as often as desired, before proceeding to the classification step.

3.3 Classification

In the classification step, the user interactively refines the results, giving feedback to the automatic classification in order to improve labeling of the sentences according to word senses. The initial automatic labels correspond to the results of the clustering phase. The user starts analyzing each sentence and decides if the automatically assigned sense label is appropriate or if a different label needs to be assigned manually. The initial default

[1] http://lucene.apache.org/solr/

[2] http://mallet.cs.umass.edu

Figure 1: Classification page

User	Assist	Accuracy	Time (mm:ss)
Annot. 1	No	0.96	8:05
Annot. 1	Yes	0.95	6:05
Annot. 2	No	0.90	10:05
Annot. 2	Yes	0.90	7:55

Table 1: Evaluation results

value for a sentence's manual label is **"unseen"**, indicating that the user has still not evaluated that sentence. Besides the available sense labels, two special labels can be assigned to a sentence: **"neither"**, to indicate that none of the available labels is applicable; **"unknown"**, meaning that the user does not know how to label it.

The classification page (figure 1 - the numbers below correspond to its elements) has a table listing each sentence with its id ①; automatically assigned sense labels in the previous ⑤ and current ⑥ iterations; confidence scores (weights) of the sense label in the previous ③ and current ④ iterations; manually assigned sense label ⑦; sentence text ⑧; and an indicator to tell if the sense label has changed between the previous and current iteration ②. The page has also a widget detailing the different word senses currently available ⑩. Besides editing the label of a sense, the user can also add new manual senses ⑪.

After modifying parameters, adding senses and manually labeling sentences, a new classification iteration can be started. The classifier uses the threshold ⑨ to identify the training data - the confidence scores are calculated by the classifier (although in the initial classification they come from the topic modeling). Furthermore, manually labeled sentences are also used as training data. We use the Naive Bayes algorithm, a classical approach for Word Sense Disambiguation, known for its efficiency (Hristea, 2013). We use the Mallet implementation of Naive Bayes, with the sentence tokens as features.

4 Evaluation

To evaluate the tool, we conducted experiments in two different scenarios: 1) using no assistive features, annotators classified sentences identifying the word senses by their own; 2) using automatically-generated clusters, annotators let the system suggest senses and then manually assigned labels to sentences, helped by the feedback-based multi-class classifier. Every annotator applied the two scenarios to a target word, namely "Galicia"[3], with three senses: a) the Spanish autonomous community, b) the region in Central-Eastern Europe; c) a football club in Brazil.

The senses were randomly distributed over 97 sentences in the first scenario (38/46/13 sentences for the respective senses) and 99 in the second (43/41/15). Sentences in both scenarios did not overlap, and were taken from a larger dataset of manually annotated sentences. The experiments were performed by two non-lexicographers (computer scientists with NLP background). We measured the time taken in every scenario and calculated the accuracy of the final results compared to the manually annotated gold standard.

Results (table 1) indicated that the time was significantly reduced when working with full assistance, compared to working without assistance. Although accuracy did vary very little, there was

[3]Although proper nouns are not present in conventional dictionaries, but rather in onomastic dictionaries, which usually do not make use of examples, it serves well, for methodological reasons, to our evaluation purposes.

a slight loss of quality in the results of the first annotator when using assistance. This might indicate a negative influence of system suggestions on the annotator or it could be attributable to a more difficult random selection of the samples in certain cases. These effects call for further investigation. However, using the tool outside the evaluation setup, we noted subjective speedups from developing smart strategies to optimize the use of the information provided by the machine (e.g. quickly annotating sentences containing specific context words or sorting by sense and confidence score). Thus, we expect the automatic assistance to have a larger impact in actual use than the present evaluation can show.

5 Conclusions and future work

We have introduced a novel system for interactive word sense disambiguation in the context of example sentences extraction. Using a combination of unsupervised and supervised methods, corpus processing and user feedback, our approach aims at helping lexicographers to properly assess and refine a list of pre-analyzed example sentences according to the senses of a target word, alleviating the burden of doing this manually. Using various state-of-the-art techniques, including clustering and classification methods for word sense induction, and incorporating user feedback into the process of shaping word senses and associating sentences to them, the tool can be a valuable addition to a lexicographer's toolset. As next steps, we plan to focus on the extraction of "good" examples, adding support for ranking sentences in different sense clusters according to operational criteria like in Didakowski et al. (2012). We also plan to extend the evaluation and to observe the strategies that users develop, in order to discover if they can inform further improvements to the system.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme (H2020-EINFRA-2014-2) under grant agreement No. 654021. It reflects only the author's views and the EU is not liable for any use that may be made of the information contained therein. It was further supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1416B (CEDIFOR) and by the German Research Foundation under grant No. EC 503/1-1 and GU 798/21-1 (INCEpTION).

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Paul Cook, Jey Han Lau, Michael Rundell, Diana McCarthy, and Timothy Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word-senses. In *Proceedings of eLex 2013*, pages 49–65, Tallinn, Estonia.

Paul Cook, Michael Rundell, Jey Han Lau, and Timothy Baldwin. 2014. Applying a word-sense induction system to the automatic extraction of diverse dictionary examples. In *Proceedings of the XVI EURALEX International Congress*, pages 319–328, Bolzano, Italy.

Jörg Didakowski, Lothar Lemnitzer, and Alexander Geyken. 2012. Automatic example sentence extraction for a contemporary german dictionary. In *Proceedings of the XV EURALEX International Congress*, pages 343–349, Oslo, Norway.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August.

Florentina T. Hristea. 2013. The naïve bayes model in the context of word sense disambiguation. In *The Naïve Bayes Model for Unsupervised Word Sense Disambiguation: Aspects Concerning Feature Selection*, pages 9–16. Springer, Heidelberg, Germany.

Adam Kilgarriff, Milo Husk, Katy McAdam, Michael Rundell, and Pavel Rychlý. 2008. Gdex: Automatically finding good dictionary examples in a corpus. In *Proceedings of the XIII EURALEX International Congress*, pages 425–432, Barcelona, Spain.

Robert Lew. 2015. Dictionaries and their users. In *International Handbook of Modern Lexis and Lexicography*. Springer, Heidelberg, Germany.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Lingmotif: Sentiment Analysis for the Digital Humanities

Antonio Moreno-Ortiz
University of Málaga
Spain
amo@uma.es

Abstract

Lingmotif is a lexicon-based, linguistically-motivated, user-friendly, GUI-enabled, multi-platform, Sentiment Analysis desktop application. Lingmotif can perform SA on any type of input texts, regardless of their length and topic. The analysis is based on the identification of sentiment-laden words and phrases contained in the application's rich core lexicons, and employs context rules to account for sentiment shifters. It offers easy-to-interpret visual representations of quantitative data (text polarity, sentiment intensity, sentiment profile), as well as a detailed, qualitative analysis of the text in terms of its sentiment. Lingmotif can also take user-provided plugin lexicons in order to account for domain-specific sentiment expression. Lingmotif currently analyzes English and Spanish texts.

1 Introduction

Lingmotif[1] is a lexicon-based Sentiment Analysis (SA) system that employs a set of lexical sources and analyzes context, by means of sentiment shifters in order to identify sentiment-laden text segments and produce two scores that qualify a text from a SA perspective. In a nutshell, it breaks down a text into its constituent sentences, where sentiment-carrying words and phrases are searched for, identified, and assigned a valence (i.e., a sentiment index). The overall score for a text is computed as a function of the accumulated negative, positive and neutral scores. Specific domains can be accounted for by applying user-provided dictionaries, which can be imported from CSV files, and used along with the application's core dictionary.

Lingmotif's SA approach could be loosely characterized as bag-of-words, since sentiment is computed solely based on the presence of certain lexical items. However, Lingmotif is not just a classifier. It also offers a visual representation of the *sentiment profile* of texts, allows to compare the profile of multiple documents side by side, and can process ordered document series. Such features are useful in discourse analysis tasks where sentiment changes are relevant, whether within or across texts, such as political speeches and narratives, or to track the evolution in sentiment towards a given topic (in news, for example).

Being focused on the end user, Lingmotif uses a simple, easy-to-use GUI that allows users to select input and options, and launch the analysis (see Figure 1). Results are generated as an HTML/Javascript document, which is saved to a predefined location and automatically sent to the user's default browser for immediate display. Internally, the application generates results as an XML document containing all the relevant data; this XML document is then parsed against one of several available XSL templates, and transformed into the final HTML.

Lingmotif is available for the Mac OS, MS Windows, and Linux platforms. It is free for non-commercial purposes.

2 Lexicon-based Sentiment Analysis

Lingmotif is a lexicon-based SA system, since it uses a rich set of lexical sources and analyzes context in order to identify sentiment laden text segments and produce two scores that qualify a text from a SA perspective. In a nutshell, it breaks down a text into its constituent sentences,

[1]This research was supported by Spain's MINECO through the funding of project Lingmotif2 (FFI2016-78141-P).

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 73–76

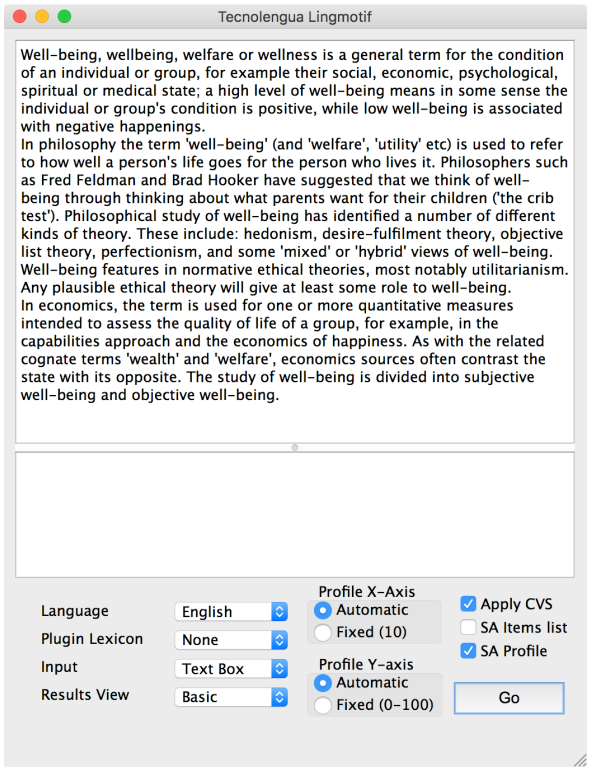


Figure 1: Lingmotifs GUI

where sentiment-carrying words and phrases are searched for, identified, and assigned a valence. For each language, Lingmotif uses a core lexicon, a set of context rules, and, optionally, one or more plugin lexicons.

In the following sections, the most salient aspects of the application's sentiment analysis engine are described. A more thorough description can be found in (Moreno-Ortiz, 2017).

2.1 Core sentiment lexicon

A lexical item in a Lingmotif lexicon can be either a single word or a multiword expression. Each entry is defined by a specification of its form, part of speech, and valence. The valence is an integer from -5 to -2 for negatives and 5 to 2 for positives. The items form can either be a literal string or a lemma. For the part-of-speech specification, Lingmotif uses the Penn Treebank tag set. A wildcard (ALL) can be used for cases where all possible parts of speech for that lemma share the same valence. Sentiment disambiguation is currently dealt with using exclusively formal features: part-of speech tags and multi-word-expressions. MWEs usually include words that may or may not have the same polarity of the expression. including such expressions can solve disambiguation for many cases. For example, we can classify as negative the word kill and then include phrases such as kill time with a neutral valence. When this is not possible, the options are to include it with the more statistically probable polarity or simply leave it out when the chances of getting the item with one polarity or another are similar.

2.2 Context rules

Context rules are Lingmotifs mechanism to deal with sentiment shifters. They work by specifying words or phrases that can appear in the immediate vicinity of the identified sentiment word. Basically, we use the same approach as (Polanyi and Zaenen, 2006). Previous implemented systems following this approach are (Kennedy and Inkpen, 2006), (Taboada et al., 2011), and (Moreno-Ortiz et al., 2010). We use simple addition or subtraction (of integers on a -5 to 5 scale in our case). When a context rule is matched, the resulting text segment is marked as a single unit and assigned the calculated valence, as specified in the rule. Lingmotifs context rules were compiled by extensive corpus analysis, studying concordances of common polarity words (adjectives, verbs, nouns, and adverbs), and then testing the rules against texts to further improve and refine them.

2.3 Plugin lexicons

Topic has been consistently shown to determine the semantic orientation of a text (Aue and Gamon, 2005), (Pang and Lee, 2008). Being a general-purpose SA system, Lingmotif provides a flexible mechanism to adapt to specific domains by means of user-provided lexicons. Lexical information contained in plugin lexicons overrides Lingmotifs core lexicon, providing domain-specific sentiment items. They can be created as a CSV file following a simple format, which is then imported into Lingmotif's internal database.

3 Single and multi-document modes

From a classification perspective, it only makes sense to use a large set of texts to be analyzed (i.e., classified). However, since Lingmotif is able to specifically identify and mark those text segments that convey sentiment, we can take advantage of this feature to measure sentiment not only in the text as a whole, but in subsections of the text, producing a *sentiment map* of the text and display the result in several ways.

A single input text can be typed or pasted in the text box area, or text files can be loaded, depending on the selected input type. Loading files allows the user to select one of them and analyze it in single-mode, or select the complete set of files.

3.1 Single-document mode

For every text analyzed, either in single or multi-document mode, Lingmotif produces a number of metrics for each individual text. The two metrics that summarize the text's overall sentiment are the Text Sentiment Score (TSS) and the Text Sentiment Intensity (TSI). Both are displayed by means of visual, animated (Javascript) gauges at the top of the results page. The numeric indexes (on a 0-100 scale) are categorized in ranges, from "extremely negative" to "extremely positive", to make numeric results more intuitively interpretable by the user. Both gauges are also color and intensity-coded in the red (negative) to green (positive) range (see Figure 2).

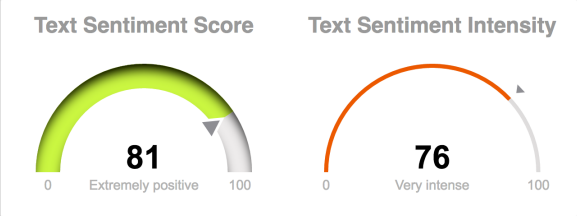


Figure 2: Sentiment scores gauges

For long texts, Lingmotif will also generate a *sentiment profile*, which is a visual representation of the text's internal structure and organization in terms of sentiment expression. This Javascript graph is interactive: hovering the data points will display the lexical items that make up that particular text segment (see Figure 3).

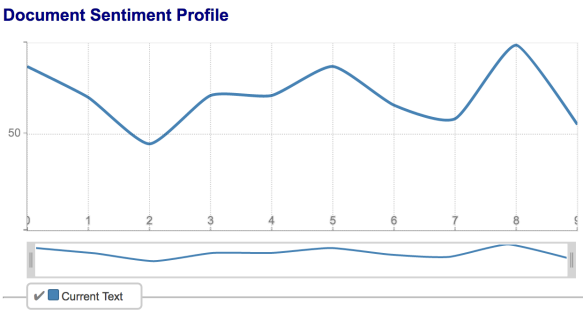


Figure 3: Document sentiment profile

The next three sections of the results document are shown in Figure 4 below. First is the quantitative data tables, which include common text metrics and a breakdown of the sentiment analysis data for the analyzed text. The final results section shows the input text after processing, where the identified sentiment items are color-coded to represent their polarity. This makes it possible to know exactly what the analyzer found in the text.

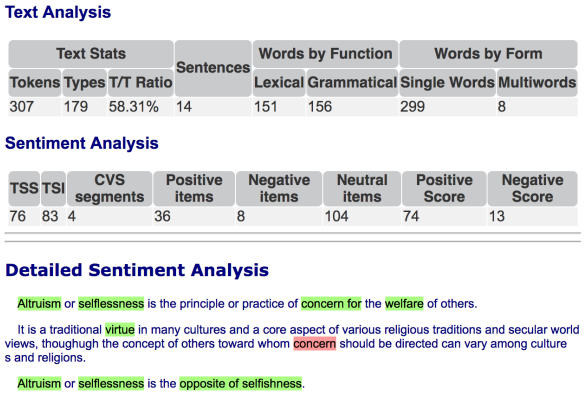


Figure 4: Sentiment scores gauges

An option exists (*advanced view*) to also show specific data for each lexical item

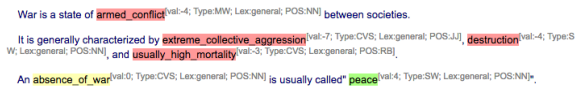


Figure 5: Detail in Advanced Mode

3.2 Multi-document analysis

Multiple input texts can be analyzed in one of several modes (see below). When in multi-document mode, Lingmotif will analyze documents one by one, generating one HTML file for each, although they will not be displayed on the browser, just saved to the output folder. When the analysis is finished, a single results page will be displayed. This page is a summary of results, and is different from the single-document results page: the gauges for TSS and TSI are now the average for the analyzed set and the detailed analysis section contains a quantitative analysis of each of the files in the set. The first column in this table shows the title of the document (file name without extension) as a hyperlink to the HTML file for that particular file.

Available multi-document analysis modes are the following:

- **Classifier** (default): a stacked bar graph and data table are offered showing classification

results based on their TSS category. The graph offers a visualization of results (see Figure 6); both its legend and the graph itself are interactive. A table summarizing the classification results is also offered.

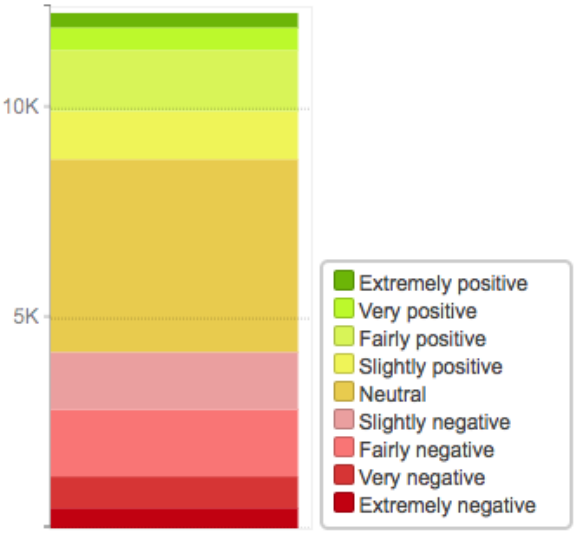


Figure 6: Classifier graph

To facilitate analysis of large sets of documents, they can be loaded from a single text file where each line is assumed to be an individual document. Lingmotif classifies documents according to their TSS, which will always include the neutral category.

- **Series**: the set of loaded files is assumed to be in order, chronological (time series) or otherwise. Each data point in the Sentiment Analysis Profile represents one document. The data point is the average TSS for that particular document.

- **Parallel**: produces a graph with one line for each file (this mode is limited to 15 documents). This is useful to compare sentiment flow in texts side by side (see Figure 7).

- **Merge**: this is a convenience option merges all loaded individual files into one single text.

4 Conclusions

Lingmotif goes beyond what SA classifiers have to offer. It offers automatic identification of sentiment-laden words and phrases, as well as text segments. Its many visual representations of the text's structure from a sentiment perspective make

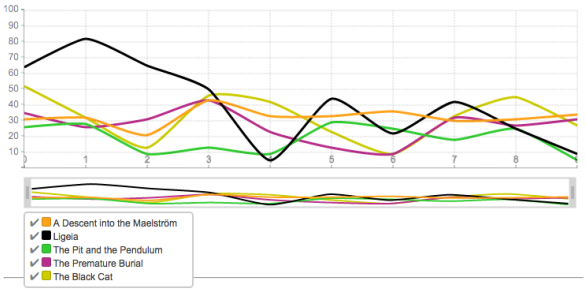


Figure 7: Multi-document analysis in parallel mode (5 documents)

it a valuable tool for research in the Digital Humanities where such tasks are relevant. The possibility to integrate user-provided lexicons enables it to adapt to any subject domain.

References

A. Aue and M. Gamon. 2005. Customizing sentiment classifiers to new domains: A case study.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence* 22(2):110–125.

A. Moreno-Ortiz. 2017. Lingmotif: A user-focused sentiment analysis tool. *Procesamiento del Lenguaje Natural* 58:21–29.

A. Moreno-Ortiz, Á. Pérez Pozo, and S. Torres Sánchez. 2010. Sentitext: sistema de análisis de sentimiento para el español. *Procesamiento del Lenguaje Natural* 45:297–298.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(12):1–135.

L. Polanyi and A. Zaenen. 2006. Contextual valence shifters. In *Computing Attitude and Affect in Text: Theory and Applications*, Springer, Dordrecht, The Netherlands, volume 20 of *The Information Retrieval Series*, pages 1–10. Shanahan, james g., qu, yan, wiebe, janyce edition.

M. Taboada, J. Brooks, M. Tofiloski, K. Voll, and M. Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics* 37(2):267–307.

RAMBLE ON: Tracing Movements of Popular Historical Figures

Stefano Menini[1-2], **Rachele Sprugnoli**[1-2], **Giovanni Moretti**[1],
Enrico Bignotti[2], **Sara Tonelli**[1] **and Bruno Lepri**[1]

[1]Fondazione Bruno Kessler / Via Sommarive 18, Trento, Italy
[2]University of Trento / Via Sommarive 9, Trento, Italy
{menini,sprugnoli,moretti,satonelli,lepri}@fbk.eu
{enrico.bignotti}@unitn.it

Abstract

We present RAMBLE ON, an application integrating a pipeline for frame-based information extraction and an interface to track and display movement trajectories. The code of the extraction pipeline and a navigator are freely available; moreover we display in a demonstrator the outcome of a case study carried out on trajectories of notable persons of the XX Century.

1 Introduction

At a time when there were no social media, emails and mobile phones, interactions were strongly shaped by movements across cities and countries. In particular, the movements of eminent figures of the past were the engine of important changes in different domains such as politics, science, and the arts. Therefore, tracing these movements means providing important data for the analysis of culture and society, fostering so-called cultural analytics (Piper, 2016).

This paper presents RAMBLE ON, a novel application that embeds Natural Language Processing (NLP) modules to extract movements from unstructured texts and an interface to interactively explore motion trajectories. In our use case, we focus on biographies of famous historical figures from the first half of the XX Century extracted from the English Wikipedia. A web-based navigator[1] related to this use case is meant for scholars without a technical background, supporting them in discovering new cultural migration patterns with respect to different time periods, geographical areas and domains of occupation. We also release the script to generate trajectories and a stand-alone version of the RAMBLE ON navigator[2], where users can upload their own set of movements taken from Wikipedia biographies.

2 Related Work

The analysis of human mobility is an important topic in many research fields such as social sciences and history, where structured data taken from census records, parish registers, mobile phones etc. are employed to quantify travel flows and find recurring patterns of movements (Pooley and Turnbull, 2005; Gonzalez et al., 2008; Cattuto et al., 2010; Jurdak et al., 2015). Other studies on mobility rely on a great amount of manually extracted information (Murray, 2013) or on shallow extraction methods. For example Gergaud et al. (2016) detect movements in Wikipedia biographies assuming that cities linked in biography pages are locations where the subject lived or spent some time.

However we believe that, even if NLP contribution has been quite neglected in cultural analytics studies, language technologies can greatly support this kind of research. For this reason, in RAMBLE ON we combine state-of-the-art Information Extraction and semantic processing tools and display the extracted information through an advanced interactive interface. With respect to previous work, our application allows to extract a wide variety of movements going beyond the birth-to-death migration that is the focus of Schich et al. (2014) or the transfers to the concentration camps of deportees during Nazism as in Russo et al. (2015).

3 Information Extraction

In Figure 1, we show the general NLP workflow behind information extraction in *RAMBLE ON*. The goal is to obtain, starting from an unstructured

[1]Available at http://dhlab.fbk.eu/rambleon/

[2]Both can be downloaded at http://dh.fbk.eu/technologies/rambleon

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 77–80

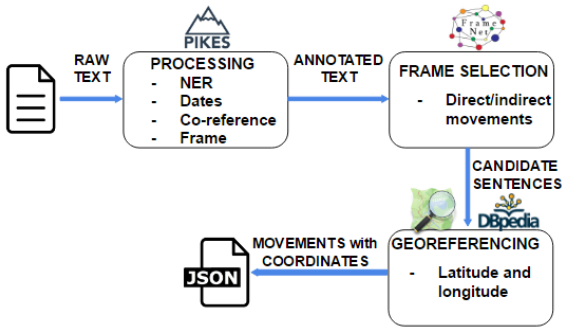


Figure 1: Information extraction workflow.

text, a set of destinations together with their coordinates and a date, each representative of the place where a person moved or lived at the given timepoint.

Input Data In our approach information extraction is performed on Wikipedia biographical pages. In the first step, these pages are cleaned up by removing infoboxes, tables and tags, keeping only the main body as raw text.

Pre-processing Raw text is processed using PIKES (Corcoglioniti et al., 2015), a suite of tools for extracting frame oriented knowledge from English texts. PIKES integrates Semafor (Das et al., 2014), a system for Semantic Role Labeling based on FrameNet (Baker et al., 1998), whose output is used to identify predicates related to movements and their arguments because its high-level organization in semantic frames is an useful way to generalize over predicates. PIKES also includes Stanford CoreNLP (Manning et al., 2014). Its modules for Named Entity Recognition and Classification (NERC), coreference resolution and recognition of time expressions are used to detect for each text: (i) mentions related to the person who is the subject of the biography; (ii) locations and organizations that can be movement destinations; (iii) dates.

Frame Selection Starting from the frames related to the *Motion* frames in FrameNet and a manual analysis of a set of biographies, we identified 45 candidate frames related to direct (e.g. *Departing*) or indirect (e.g. *Residence*) movements of people. After a manual evaluation of these 45 frames on a set of biographies annotated with PIKES, we removed 16 of them from the list of candidate frames because of the high number of false positives. These include for example *Escaping*, *Getting_underway* and *Touring*. Combining the information from the CoreNLP modules in PIKES with the remaining 29 frames listed in Table 1, our application extracts a list of candidate sentences, containing a date and a movement of the subject together with a destination. These represent the geographical position of a person at a certain time.

FRAME
Arriving
Attending
Becoming_a_member
Being_employed
Bringing
Cause_motion
Colonization
Come_together
Conquering
Cotheme
Departing
Detaining
Education_teaching
Fleeing
Inhibit_movement

FRAME
Meet_with
Motion
Receiving
Residence
Scrutiny
Self_motion
Sending
Speak_on_topic
State_continue
Temporary_stay
Transfer
Travel
Used_up
Working_on

Table 1: List of frames selected for RAMBLE ON[4]

Georeferencing To georeference all the destinations mentioned in the candidate sentences RAMBLE ON uses Nominatim[5]. Due to errors by the NERC module (e.g., *Artaman League* annotated as geographical entity), some destinations can lack coordinates and thus are discarded. Moreover, for each biography, the places and dates of birth and death of the subject are added as taken from DBpedia.

Output Data Details about the movements extracted from each Wikipedia biography, e.g. date, coordinates and the original snippet of text, are saved in a JSON file as shown in the example below. This output format accepts additional fields with information about the subject of the biography, e.g. gender, occupation domain, that could be extracted from other resources.

```
"name": "Maria_Montessori",
"movements": [{
 "date": 19151100,
 "place": "Italy",
 "latitude": 41.87194,
 "longitude": 12.5673,
 "predicate_id": "t3147",
 "predicate": "returned",
 "resource": "FrameNet",
 "resource_frame": "Arriving",
 "place_frame": "@Goal",
 "snippet": "Montessori's father died in
 November 1915, and she returned to Italy."
}]
```

[5] https://nominatim.openstreetmap.org/

4 Ramble On Navigator

Movements as extracted with the procedure described in Section 3 are graphically presented on an interactive map that visualizes trajectories between places. The interface, called `RAMBLE ON Navigator`, is built using technology based on web standards (HTML5, CSS3, Javascript) and open source libraries for data visualization and geographical representation, i.e. d3.js and Leaflet. Through this interface, see Figure 2, it is possible to filter the movements on the basis of the time span or to search for a specific individual. Moreover, if information about nationality and domain of occupation is provided in the JSON files, the Navigator allows to further filter the search. Hovering the mouse on a trajectory, the snippet of text from which it was automatically extracted appears on the bottom left. Information about all the movements related to a place is displayed when hovering on a spot on the map. The trajectories have an arrow indicating the route destination and are dashed if the movement described by the snippet is started before the selected time span.
The online version of the `Navigator` shows the output of the case study presented in Section 5, while the stand-alone application also allows to upload another set of data.

5 Case Study

We relied on the Pantheon dataset (Yu et al., 2016) to identify a list of notable figures to be used in our case study. We chose Pantheon since it provides a ready-to-use set of people already classified into categories based on their domain occupation (e.g., *Arts*, *Sports*), birth year, nationality and gender. More specifically, we considered 2,407 individuals from Europe and North America living between 1900 and 1955. First we downloaded the corresponding Wikipedia pages, as published in April 2016, collecting a corpus of more than 7,5 million words. Then we used the workflow described in Section 3 and we enriched output data with the categories taken from Pantheon. We manually refined the output by removing the sentences wrongly identified as movements (14.02%), for example those not referring to the subject of the biography (e.g., *When communist North Korea invaded South Korea in 1950, he sent in U.S. troops*). The final dataset resulted in 2,929 sentences from 1,283 biographies, since 1,124 individuals had no associated movements. This may be due to either an actual lack of sentences concerning movements or errors in the automatic processing, e.g., missed identification of places or dates. Table 2 shows the distribution per domain of the individuals with associated movements. Moreover, we corrected the coordinates of places wrongly georeferenced (6.7%). The extracted movements are evoked by predicates associated to 66 different lemmas (lexical units in FrameNet). The most frequent lemmas ($>$ 100 occurrences) are: *return* (567), *move* (556), *visit* (253), *travel* (188), *attend* (182), *go* (153), *live* (111), *arrive* (107).

DOMAIN	# of individuals	# of movements
Arts	647 (348)	788
Science & Technology	591 (318)	631
Humanities	502 (276)	709
Institutions	483 (255)	633
Public Figure	69 (30)	55
Sports	59 (30)	54
Business & Law	38 (13)	22
Exploration	18 (13)	37
TOTAL	2,407 (1,283)	2,929

Table 2: Domain distribution of individuals in our use case. In brackets the number of individuals with movements.

6 Conclusion and Future Work

We presented an automatic approach for the extraction and visualisation of motion trajectories, which is easy to extend to different datasets, and that can provide insights for studies in many fields, e.g., history and sociology.

In the future, we will mainly focus on improving the system coverage. Currently, missing trajectories are mainly due to (i) the presence of predicates not recognized as lexical units in FrameNet, e.g. *exile*; (ii) the lack of information in the English Wikipedia biography, and (iii) the presence of sentences with complex temporal structures, e.g., *Cummings returned to Paris in 1921 and remained there for two years before returning to New York*. These issues can be dealt with by adding missing predicates to FrameNet, extend Pikes to other languages and experimenting with different systems for temporal information processing (Llorens et al., 2010). We also plan to apply the methodology presented in (Aprosio and Tonelli, 2015) to automatically recognize the Wikipedia text passages dealing with biographical information, so to discard sections containing useless information.

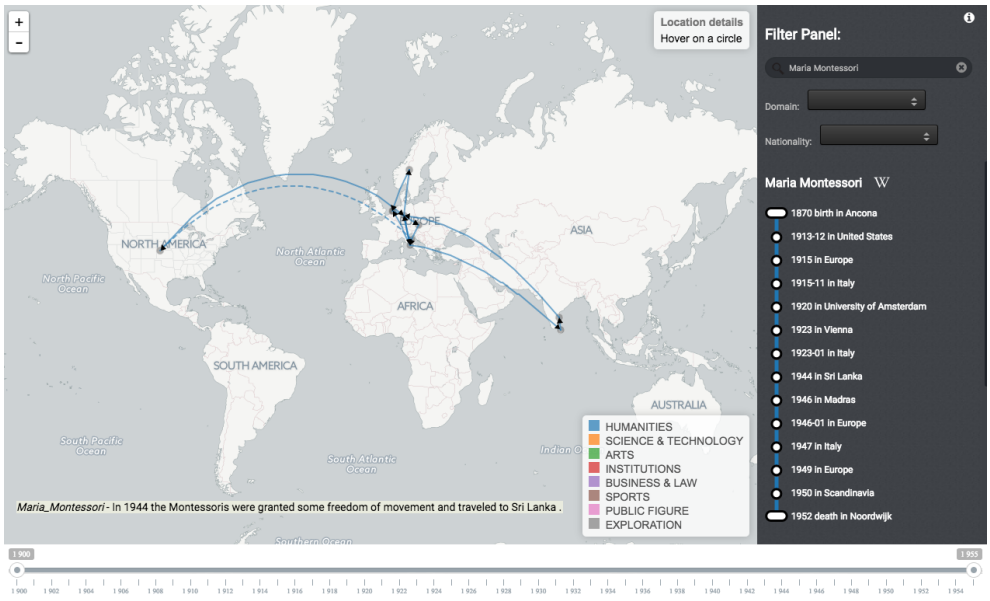


Figure 2: Screenshot of Ramble On Navigator.

References

Alessio Palmero Aprosio and Sara Tonelli. 2015. Recognizing Biographical Sections in Wikipedia. In *Proceedings of EMNLP 2015*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL '98*, pages 86–90.

Ciro Cattuto, Wouter Van den Broeck, Alain Barrat, Vittoria Colizza, Jean-François Pinton, and Alessandro Vespignani. 2010. Dynamics of person-to-person interactions from distributed RFID sensor networks. *PloS one*, 5(7):e11596.

Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio. 2015. Extracting Knowledge from Text with PIKES. In *Proceedings of the International Semantic Web Conference (ISWC)*.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Olivier Gergaud, Morgane Laouénan, Etienne Wasmer, et al. 2016. A brief history of human time: Exploring a database of'notable people'. Technical report, Sciences Po Departement of Economics.

Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782.

Raja Jurdak, Kun Zhao, Jiajun Liu, Maurice AbouJaoude, Mark Cameron, and David Newth. 2015. Understanding human mobility from Twitter. *PloS one*, 10(7):e0131469.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (English and Spanish): Evaluating CRFs and Semantic Roles in Tempeval-2. In *Proceedings of the 5th SemEval*, pages 284–291.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Sarah Murray. 2013. Spatial Analysis and Humanities Data: A Case Study from the Grand Tour Travelers Project. In *A CESTA Anthology*. Stanford.

Andrew Piper. 2016. There will be numbers. *CA: Journal of Cultural Analytics*, 1(1).

Colin Pooley and Jean Turnbull. 2005. *Migration and mobility in Britain since the eighteenth century*. Routledge.

Irene Russo, Tommaso Caselli, and Monica Monachini. 2015. Extracting and Visualising Biographical Events from Wikipedia. In *Proceedings of the 1st Conference on Biographical Data in a Digital World*.

Maximilian Schich, Chaoming Song, Yong-Yeol Ahn, Alexander Mirsky, Mauro Martino, Albert-László Barabási, and Dirk Helbing. 2014. A network framework of cultural history. *Science*, 345(6196):558–562.

Amy Zhao Yu, Shahar Ronen, Kevin Hu, Tiffany Lu, and César A. Hidalgo. 2016. Pantheon 1.0, a manually verified dataset of globally famous biographies. *Scientific data*, 3.

Autobank: a semi-automatic annotation tool for developing deep Minimalist Grammar treebanks

John Torr
School of Informatics
University of Edinburgh
11 Crichton Street, Edinburgh, UK
john.torr@cantab.net

Abstract

This paper presents Autobank, a prototype tool for constructing a wide-coverage Minimalist Grammar (MG) (Stabler, 1997), and semi-automatically converting the Penn Treebank (PTB) into a deep Minimalist treebank. The front end of the tool is a graphical user interface which facilitates the rapid development of a seed set of MG trees via manual reannotation of PTB preterminals with MG lexical categories. The system then extracts various dependency mappings between the source and target trees, and uses these in concert with a non-statistical MG parser to automatically reannotate the rest of the corpus. Autobank thus enables deep treebank conversions (and subsequent modifications) without the need for complex transduction algorithms accompanied by cascades of ad hoc rules; instead, the locus of human effort falls directly on the task of grammar construction itself.

1 Introduction

Deep parsing techniques, such as CCG parsing, have recently been shown to yield significant benefits for certain NLP applications. However, the construction of new treebanks for training and evaluating parsers using different formalisms is extremely expensive and time-consuming. The Penn Treebank (PTB) (Marcus et al., 1993), for instance, the most commonly used treebank within NLP, took a team of linguists around three years to develop. Its structures were loosely based on Chomsky's Extended Standard Theory (EST) from the 1970s and, nearly half a century on, these look very different from contemporary Chomskyan Minimalist analyses. Considerable theoretical advances have been made during that time, including the discovery of many robust cross-linguistic generalizations. These could prove very useful for NLP applications such as machine translation, particularly with respect to under-resourced languages. Unfortunately, the lack of any Minimalist treebank to date has meant that there has been very little research into statistical Minimalist parsing (though see Hunter and Dyer (2013)).

Given the labour intensity of constructing new treebanks from scratch, computational linguists have developed techniques for converting existing treebanks into different formalisms (e.g. Hockenmaier and Steedman (2002), Chen et al. (2006)). These approaches generally involve two main subtasks: the first is to create a general algorithm to translate the existing trees into the representational format of the target formalism, for example by binarizing and lexicalizing the source trees and, in the case of CCGbank (Hockenmaier and Steedman, 2002), replacing traces of movement with alternative operations such as type-raising and composition; the second task involves coding cascades of ad hoc rules to non-trivially modify and/or enrich the underlying phrase structures, either because the target formalism requires this, or because the researcher disagrees with certain theoretical decisions made by the original treebank's annotators. CCGbank, for instance, replaces many small clauses in the PTB by a two-complement analysis following Steedman (1996).

Autobank is a new approach to semi-automatic treebank conversion. It was designed to avoid the need for coding complex transduction algorithms and cascades of ad hoc rules. Such rules become far less feasible (and difficult for future researchers to modify) when transducing to a very deep formalism such as a Minimalist Grammar (MG) (Stabler, 1997), whose theory of phrase

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 81–86

structure[1] differs considerably from that of the PTB[2]. Furthermore, given the many competing analyses for any given construction in the Minimalist literature, no single MG treebank will be universally accepted. It is therefore hoped that Autobank will stimulate wider interest in broad coverage statistical Minimalist parsing by providing researchers with a relatively quick and straightforward way to engineer their own MGs and treebanks, or to modify existing ones.

Autobank works as follows: the PTB first undergoes an initial preprocessing phase. Next, the researcher builds a seed MG corpus by annotating lexical items on PTB trees with MG categories and then selecting from among a set of candidate parses which are output using these categories by MGParse, an Extended Directional Minimalist Grammar (EDMG) parser (see Torr and Stabler (2016)). The system then extracts various dependency mappings between the source and target trees. Next, a set of candidate parses is generated for the remaining trees in the PTB, and these are scored using the dependency mappings extracted from the seeds. In this way, the source corpus effectively adopts a disambiguation role in lieu of any statistical model. The basic architecture of the system, which was implemented in Python and its Tkinter module, is given in fig 1.

2 Preprocessing

Autobank includes a module for preprocessing the PTB which corrects certain mistakes and adds some additional annotations carried out by various researchers since the treebank's initial release. For example, following Hockenmaier and Steedman (2002), we have corrected cases where verbs were incorrectly labelled with the past tense tag VBD instead of the past participle tag VBN. We also extend the PTB tag set to include person, number and gender[3] information on nominals and pronominals, in order to constrain reflexive binding and agreement phenomena in MGbank.

The semantic role labels of PropBank (Palmer et al., 2005) and Nombank (Meyers et al., 2004) have also been added onto PTB non-terminals[4], along with the head word and its span. For instance, the AGENT subject NP *Jack* in the sentence, *Jack helped her*, would be annotated with the tag ARG0`{helped<1,2>}`. We have also added the additional NP structure from Vadas and Curran (2007), the additional structure for hyphenated compounds included in the Ontonotes 5 (Weischedel et al., 2012) version of the PTB, and the additional structure and role labels for coordination phrases recently released by Ficler and Goldberg (2016). For all structure added, any function tags are redistributed accordingly[5].

Finally, PropBank includes additional antecedent-trace co-indexing which we have also imported, and some of this implies the need for additional NP structure beyond what Vadas and Curran have provided. For instance, in the phrase, *the unit of New York-based Lowes Corp that *T* makes kent cigarettes*, the original annotation has *the unit* and *of New York-based Lowes Corp* as separate sister NP and PP constituents (with an SBAR node sister to both), both of which are co-indexed with the subject trace (**T**) position in PropBank. In such cases we have added an additional NP node resolving the two constituents into a single antecedent NP.

3 The manual annotation phase

Autobank provides a powerful graphical user interface enabling the researcher to construct an (ED)MG by relabelling PTB preterminals with MG categories and selecting the correct MG parse from a set of candidates generated by the parser.

The main annotation environment is shown in fig 2. The PTB tree and its MG candidates are respectively displayed on the top and bottom of the screen. Between these are a number of buttons allowing for easy navigation through the PTB, including a regular expression search facility for locating specific construction types by searching both the bracketing and the string. The user can also choose to focus on sentences of a given string length. On the left, the sentence is displayed from top to bottom, each word with a drop-down menu listing all MG categories so far associated with that word's PTB preterminal category. Like

[1] MGs are a mildly context sensitive and computational interpretation of Chomsky's (1995) Minimalist Program.

[2] For example, Minimalist trees contain many more null heads and traces of phrasal movement, along with shell and Xbar phrase structures, cartographic clausal and nominal structures, functional heads, head movements, covert movements/Agree operations etc.

[3] We used the NLTK name database to derive gender on proper nouns.

[4] Among other things, these crucially distinguish adjuncts from arguments, raising/ECM from subject/object control, and *promise*-type subject control from object control/ECM.

[5] We use a modified version of Collins' (1999) head finding rules for this task as well as for the dependency extraction.

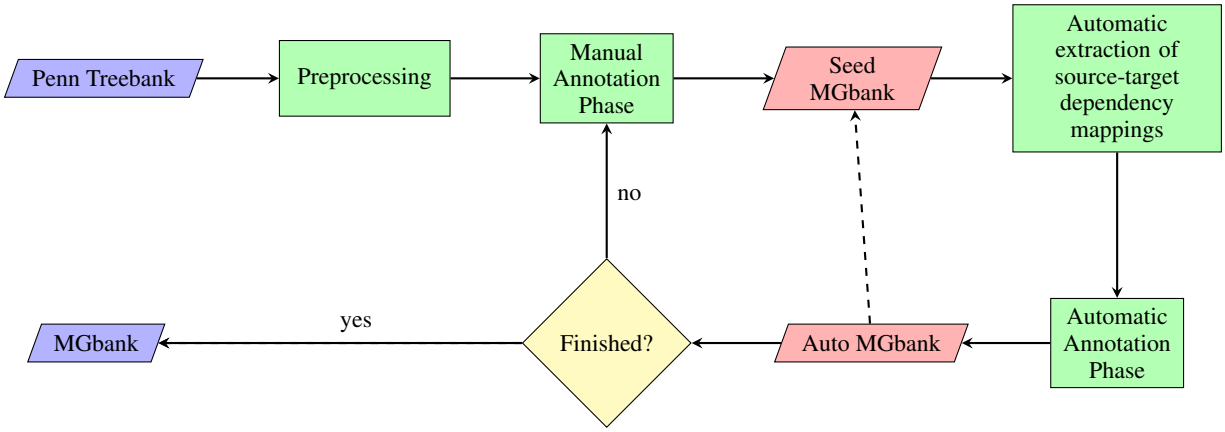


Figure 1: Autobank architecture.

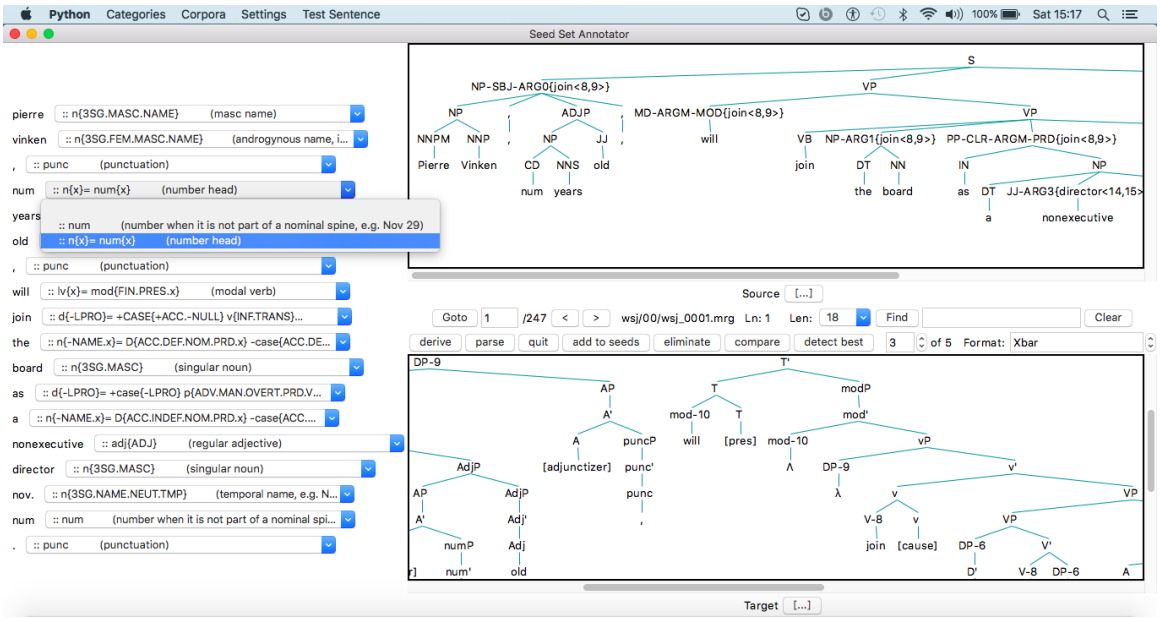


Figure 2: The main annotation environment.

CCG, EDMG is a strongly lexicalized and derivational formalism, with all subcategorization and linear order information encoded on lexical items. EDMG categories are sequences of features ordered from left to right which are checked and deleted as the derivation proceeds. For example, the hypothetical category, d= =d v[6], could be used to represent a transitive verb that first looks to its right for an object with d as its first feature (i.e. a DP), before selecting a DP subject on its left and thus yielding a constituent of category v, i.e. a VP.

MGParse category features can also include subcategorization and agreement properties and requirements, and allow for percolation of such features up the tree via a simple unification mechanism[7]. In lieu of any statistical model, this enables the human annotator to tightly constrain the grammar and thus reduce the amount of local and global ambiguity present during manual and automatic annotation. For example, the category: v{+TRANS.x}= +case{+ACC.-NULL} =d lv{TRANS.x}, could be used for the null causative light verb (so-called *little v*) that is standardly assumed in Minimalism to govern a main transitive verb. It specifies that its VP complement must have the property TRANS and that the ob-

[6]This category (which is actually used for ditransitives by MGParse) is similar to the CCG category (S\NP)/NP.

[7]As in CCG, such unification is limited to atomic property values rather than the sorts of unbounded feature structures found in Head-Driven Phrase Structure Grammars; unification here is therefore not re-entrant.

ject whose case feature it will check (in this case via covert movement) must have the property ACC but must not have the property NULL, and that following selection of an AGENT DP specifier the resulting vP will have the property TRANS. Furthermore, any additional properties carried by the VP complement (such as PRES, 3SG etc.) will be percolated onto the vP (or rather onto its selectee (lv) feature) owing to the *x* variable.

Both overt and null (i.e. phonetically silent) MG categories can be added to the system and later modified using the menu system at the top of the screen. Any time the user attempts to modify a category, the system will first reparse any trees in the seed set containing that category to ensure that the same Xbar tree can still be generated for the sentence in question following the modification[8].

Once the user has selected an MG category for each word in the sentence, clicking `parse` causes MGParse to return all possible parses using these categories. Parsing without annotating some or all of the words in the sentence is also possible: MGParse will simply try all available MG categories already associated with a word's PTB preterminal in the seed set. Once returned, the trees can be viewed in several formats, including multiple MG derivation tree formats, and Xbar tree and MG (bare phrase structure) derived tree formats. Candidate parses can be viewed side-by-side for comparison, and there is the option to perform a diff on the bracketings, and to eliminate incorrect candidates from consideration. Once the user has identified the correct tree, they can click `add to seeds` to save it; seeds can be viewed and removed at any point using the native file system.

There will inevitably be occasions when the parser fails to return any parses. In these cases it is useful to build up the derivation step-by-step to identify the point where it fails, and Autobank provides an interface for doing just this (see fig 3). Whereas in annotation mode null heads were kept hidden for simplicity, in derivation mode the entire null lexicon is available, along with the overt categories the user selected on the main annotation screen. Other features of the system include a test sentence mode, a corpus stats display, a facility for automatically detecting the best candidate MG parse (to test the performance of the automatic annotator, and speed up annotation), a parser settings menu, and an option for backing up all data.

The extreme succinctness of the (strongly) lexicalized (ED)MG formalism, as discussed in Stabler (2013), means that seed set creation can be a relatively rapid process. Like CCGs, MGs have abstract rule schemas which generalize across categories, thus dramatically reducing the size of the grammar. Taking CCGbank's 1300 categories as an approximate upper bound, a researcher working five days a week on the annotation process and adding 20 MG categories per day to the system should have added enough MG categories to parse all the sentences of the PTB within 3 months.

4 The automatic annotation phase

Once the user has created an initial seed set, they can select `auto generate corpus` from the `corpus` menu. This prompts the system to extract a set of dependency mappings and lexical category mappings holding between each seed MG tree and its PTB source tree. To achieve this, the system traverses the PTB and MG trees and extracts a Collins-style dependency tuple for every non-head child of every non-terminal in each tree. The tuples include the head child and non-head child categories, the parent category, any relevant function tags, the directionality of the dependency, and the head child's and non-head child's head words and spans. Where there are multiple tuples in a tree with the same head and non-head word spans, these are grouped together into a *chain*.

For example, for the sentence *the doctor examined Jack*, the AGENT subject NP in a PTB-style tree would yield the following dependency: [VP, *examined*, `<2,3>`, NP, *doctor*, `<1,2>`, S, [ARG0, SUBJ], left]. Many Minimalists assume that AGENT subjects are base-generated inside the verb phrase (Koopman and Sportiche, 1991) in spec-vP, before moving to the surface subject position in spec-TP. Although in its surface position the subject is a dependent of the T(ense) head, it is the lexical verb which is the semantic head of the extended projection (i.e. of the CP clause containing it), and both syntactic and semantic heads are used here when generating the tuples[9]. Two tuples are therefore extracted for the dependency

[8] In general, subcategorization properties and requirements are the only features on an MG category that can be modified without first removing all seed parses containing that category as they do not affect the Xbar tree's geometry (though they can license or prevent its generation).

[9] This also allows the system to capture certain systematic changes in constituency and recognize, for instance, that temporal adjuncts tagged with a TMP label and attaching to VP in the PTB should attach to TP in the MG tree.

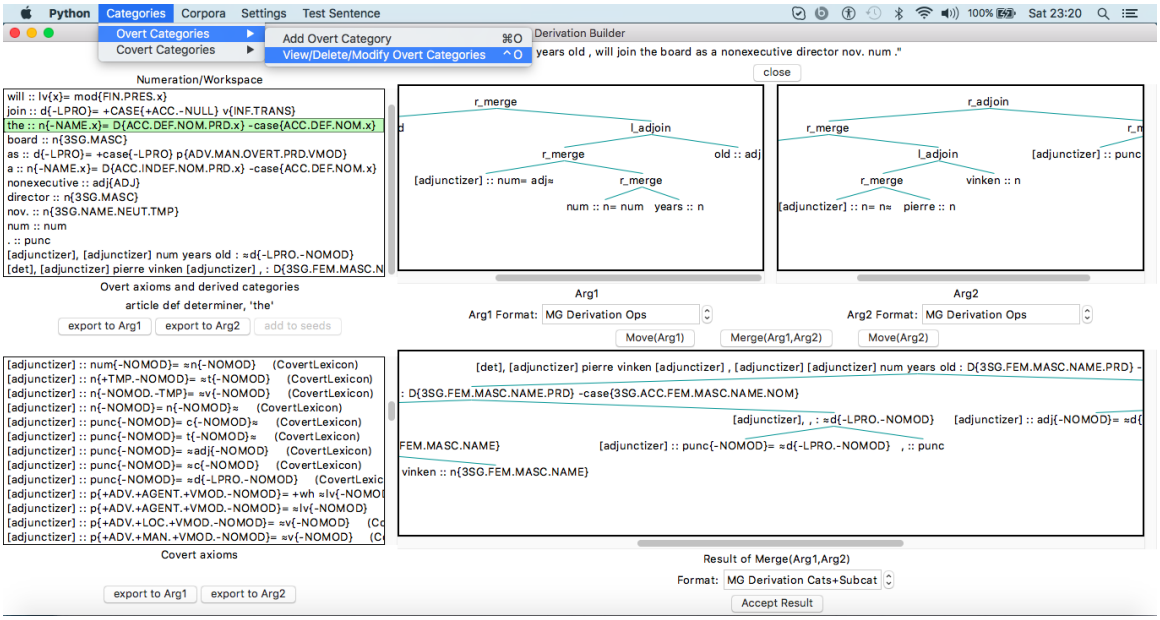


Figure 3: The step-by-step derivation builder.

between the verb and the subject and together they form a chain representing the latter's deep and surface positions. The system then establishes mappings from PTB tuples/chains to MG tuples/chains which share the same head and non-head spans (this includes instances where the relation between the head and dependent has been reversed).

Each mapping is stored in three forms of varying degrees of abstraction. In the first, all word-specific information (i.e. the head and non-head words and their spans) is deleted; in the second the spans and non-head word are deleted, but (a lemmatized version of) the head word is retained, while in the third the spans are deleted but both the (lemmatized) head and non-head words are retained. The fully reified mapping for our subject dependency, for instance, would be: [VP, *examine*, NP, *doctor*, S, [ARG0, SUBJ], left] —> [[T', *examine*, DP, *doctor*, TP, left], [v', *examine*, DP, *doctor*, vP, left]]. Including both abstract and reified mappings allows the system to recognise not only general phrase structural correspondences, but also more idiosyncratic mappings conditioned by specific lexical items, as in the case of idioms and light verb constructions, for instance.

The system next parses the remaining sentences, selecting a set of potential MG categories for each word in each sentence using the lexical category mappings[10]. Whenever a dependency mapping is discovered which has previously been seen in the seeds, the MG tree containing it is awarded with a point; the candidate with the most points is added to the Auto MGbank. The abstract mapping above, for instance, ensures that for transitive sentences, trees containing the subject trace in spec vP are preferred. The user can choose to specify the number and maximum string length of the trees that are automatically generated (together with a timeout value for the parser) and can then inspect the results and transfer any good trees into the seed corpus, thereby rapidly expanding it.

5 Conclusion

Autobank is a GUI tool currently being used to semi-automatically construct MGbank, a deep Minimalist version of the PTB. Minimalism is a lively theory, however, and in continual flux. Autobank was therefore designed with reusability in mind, in the hope that other researchers will use it to create alternative Minimalist treebanks, either from scratch or by modifying an existing one, and to stimulate greater interest in computational Minimalism and statistical MG parsing. The system could also potentially be adapted for use with other source and (lexicalised) target formalisms.

[10]Note that there will be many MG categories for every PTB category, which will make parsing quite slow for certain sentences. To ameliorate this, once enough seeds have been added, an MG supertagger (see Lewis and Steedman (2014)) will be trained and used to reduce the amount of lexical ambiguity; to improve things further, multiple sentences will be processed in parallel during automatic annotation.

Acknowledgements

Many thanks to Ed Stabler for providing the initial inspiration for this project, and for his subsequent guidance and encouragement last summer. The work described in this paper was funded by Nuance Communications Inc. and the Engineering and Physical Sciences Research Council. I would also like to thank the reviewers for their comments and also my supervisors Mark Steedman and Shay Cohen for their help and guidance during the development of the Autobank system and the writing of this paper.

References

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2006. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jessica Ficler and Yoav Goldberg. 2016. Coordination annotation extension in the penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, German, Volume 1: Long Papers*, pages 834–842.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1974–1981.

Tim Hunter and Chris Dyer. 2013. Distributions on minimalist grammar derivations. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 1–11, Sofia, Bulgaria, August. The Association of Computational Linguistics.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85(2-3):211–258.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with semi-supervised supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Mitch Marcus, Beatrice Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proceedings of HLT-EACL Workshop: Frontiers in Corpus Annotation*.

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

Edward Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics (LACL'96)*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, New York. Springer.

Edward Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5:611–633.

Mark Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monograph 30. MIT Press, Cambridge, MA.

John Torr and Edward P. Stabler. 2016. Coordination in minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the Twelfth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+12)*, pages 1–17.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, June. ACL.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D. Hwang, Claire Bonial, Jinho Choi, Aous Mansouri, Maha Foster, Abdel aati Hawwary, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston. 2012. *OntoNotes Release 5.0 with OntoNotes DB Tool v0.999 beta*.

Chatbot with a Discourse Structure-Driven Dialogue Management

Boris Galitsky
Knowledge Trail Inc.
San Jose, CA, USA
bgalitsky@hotmail.com

Dmitry Ilvovsky
National Research University
Higher School of Economics
Moscow, Russia
dilvovsky@hse.ru

Abstract

We build a chat bot with iterative content exploration that leads a user through a personalized knowledge acquisition session. The chat bot is designed as an automated customer support or product recommendation agent assisting a user in learning product features, product usability, suitability, troubleshooting and other related tasks. To control the user navigation through content, we extend the notion of a linguistic discourse tree (DT) towards a set of documents with multiple sections covering a topic. For a given paragraph, a DT is built by DT parsers. We then combine DTs for the paragraphs of documents to form what we call extended DT, which is a basis for interactive content exploration facilitated by the chat bot. To provide cohesive answers, we use a measure of rhetoric agreement between a question and an answer by tree kernel learning of their DTs.

1 Introduction

Modern search engines have become very good at understanding typical, most popular user intents, recognizing topic of a question and providing a relevant links. However, search engines are not necessarily capable of providing an answer that would match a style, personal circumstances, knowledge state, an attitude of a user who formulated a query. This is particularly true for long, complex queries, and for a dialogue-based type of interactions. In a chat bot, the flow *query – clarification request - clarification response - candidate answer* should be cohesive, not just maintain a topic of a conversation. Moreover, modern search engines and modern chat bots are unable to leverage an immediate, explicit user feedback on what is most interesting and relevant to them.

The chat bot we introduce in this demo paper is inspired by the idea that knowledge exploration should be driven by navigating a single discourse tree (DT) built for the whole corpus of relevant content. We refer to such tree as *extended discourse tree*. Moreover, to select rhetorically cohesive answers, chat bot should be capable of classifying question-answer pairs as cohesive or not. This can be achieved by learning the pairs of extended discourse trees for the question-answer pairs. A question can have an arbitrary rhetoric structure as long as the subject of this question is clear to its recipient. An answer on its own can have an arbitrary rhetoric structure. However, these structures, the DT of a question and the DT of an answer should be correlated when this answer is appropriate to this question. We apply a computational measure for how logical, rhetoric structure of a request or a question is in agreement with that of a response, or an answer.

Over last decade, Siri for iPhone and Cortana for Windows Phone have been designed to serve as digital assistants. They analyze input question sentences and return suitable answers for users queries (Crutzen et al., 2011). However, they assume patterned word sequences as input commands. Moreover, there are previous studies that combine natural language processing techniques with ontology technology to implement computer system for intellectual conversation. There are chat bot systems including ALICE3, which utilizes an ontology, like Cyc, API.ai and Amazon Lex, the platforms for developers to build chat bots. Most of these systems expected the correct formulation of a question, certain domain knowledge and a rigid grammatical structure in user query sentences; however, less rigid structured sentences can appear in a users utterance to a chat bot. Developers of chat bot platforms

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 87–90

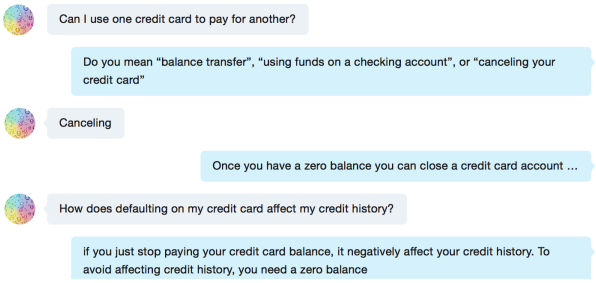


Figure 1: Sample dialogue. User questions and responses are aligned on the left and system responses - on the right.

can specify preset Q/A pairs, but are unable to introduce domain-independent rhetoric constraints which are much more flexible and reusable.

2 Personalized and Interactive Domain Exploration Scenarios

Most chat bots are designed to imitate human intellectual activity maintaining a dialogue. The purpose of the chat bot with iterative exploration is to provide most efficient and effective information access for users. A vast majority of chat bots nowadays, especially based on deep learning, try to build a plausible sequence of words (Williams, 2012) to serve as an automated response to user query. Such most plausible responses, sequences of syntactically and semantically compatible words, are not necessarily most informative. Instead, in this demo we focus on a chat bot that helps a user to navigate to the exact, insightful answer as fast as possible.

For example, if a user is formulated her query *Can I use one credit card to pay for another*, the chat bot attempts to recognize a user intent and a background knowledge about this user to establish a proper context. The chat bot leverages an observation learned from the web that an individual would usually want *to pay with one credit card for another to avoid late payment fee when cash is unavailable* as long as the user does not specify her other circumstances.

To select a suitable answer from a search engine, a user first reads snippets one-by-one and then proceeds to the linked document to consult in detail. Reading the answer #n does not usually help to decide which next answer should be consulted, so a user proceeds is to answer #n+1. Since answers are sorted by popularity, for a given user there is no better way than just proceed from

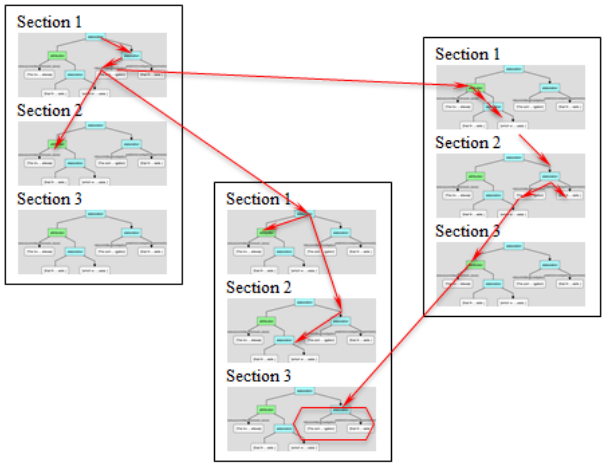


Figure 2: Extended discourse tree for three documents used to navigate to a satisfactory answer

top to bottom on the search results page. On the contrary, the chat bot allows a convergence in this answer navigation session since the answer #n+1 is suggested based on additional clarification submitted after the answer #n is consulted by the user. Chat bot provides *topics* of answers for a user to choose from. These topics give the user a chance to assess how her request was understood on one hand and restore the knowledge area associated with her question on the other hand. In our examples, topics include *balance transfer*, *using funds on a checking account*, or *canceling your credit card*. A user is prompted to select a clarification option, drill into either of these options, or decline all options and request a new set of topics which the chat bot can identify.

3 Navigation with the Extended DT

On the web, information is usually represented in web pages and documents, with certain section structure. Answering questions, forming topics of candidate answers and attempting to provide an answer based on user selected topic are the operations which can be represented with the help of a structure that includes the DTs of texts involved. When a certain portion of text is suggested to a user as an answer, this user might want to drill in something more specific, ascend to a more general level of knowledge or make a side move to a topic at the same level. These user intents of navigating from one portion of text to another can be represented in most cases as coordinate or subordinate discourse relations between these portions.

Rhetorical Structure Theory (RST), proposed

by (Mann and Thompson, 1988), became popular as a framework for parsing the discourse relations and discourse structure of a text, represents texts by labeled hierarchical structures such as DTs. A DT expresses the author's flow of thoughts at the level of a paragraph or multiple paragraphs. But DTs become fairly inaccurate when applied to larger text fragments, or documents. To enable a chat bot with a capability to form and navigate a DT for a corpus of documents with sections and hyperlinks, we introduce the notion of ofextended DT. To avoid inconsistency we would further refer to the original DTs as conventional. To construct conventional discourse trees we used one of the existing discourse parsers (Joty et al., 2013).

The intuition behind navigation is illustrated in Fig.2. Three areas in this chart denote three documents each containing three section headers and section content. For section content, discourse trees are built for the text. A navigation starts with the route node of a section that matches the user query most closely. Then the chat bot attempts to build a set of possible topics, which are the satellites of the root node of the DT. If the user accepts a given topic, the navigation continues along the chosen edge. Otherwise, when no topic covers the user interest, the chat bot backtracks the DT and proceeds to the other section (possibly of another documents) which matched the original user query second best. Finally, the chat bot arrives at the DT node where the user is satisfied (shown by hexagon) or gives up and starts a new exploratory session . Inter-document and inter-section edges for relations such as *elaboration* play similar role in knowledge exploration navigation to internal edges of a conventional discourse tree.

4 Relying on Q/A rhetoric agreement to select the most suitable answers

In conventional search approach, as a baseline, Q/A match is measured in terms of keyword statistics such as TF*IDF. To improve search relevance, this score is augmented by item popularity, item location or taxonomy-based score (Galitsky et al., 2013; Galitsky et al., 2014). The feature space includes Q/A pairs as elements, and a separation hyper-plane splits this feature space. We combine DT(Q) with DT(A) into a single tree with the root Q/A pair (Fig. 3). We then classify such pairs into correct (with high agreement) and incorrect (with low agreement). Having multiple answer

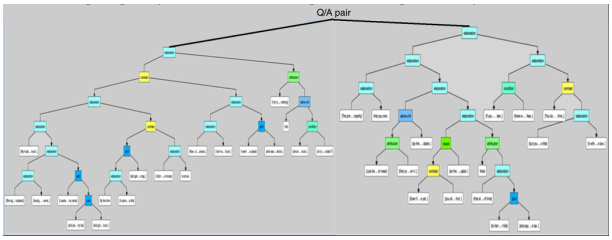


Figure 3: Forming the Request-Response pair as an element of a training set

candidates, we select those with higher classification score. Classification algorithm is based on the tree kernel learning on conventional discourse trees. Their regular nodes are rhetoric relations, and terminal nodes are elementary discourse units (phrases, sentence fragments) which are the subjects of these relations.

We selected Yahoo! Answer set of question-answer pairs with broad range of topics. Out of the set of 4.4 million user questions we selected 20000 which included more than two sentences. Answers for most of the questions are fairly detailed so no filtering was applied to answers. There are multiple answers per questions and the best one is marked. We consider the pair *Question - Best Answer* as an element of the positive training set and *Question - Other Answer* as the one of the negative training set. To derive the negative training set, we either randomly selected an answer to a different but somewhat related question, or formed a query from the question and obtained an answer from web search results.

5 Evaluation

We compare the efficiency of information access using the proposed chat bot with a major web search engines such as Google, for the queries where both systems have relevant answers. For a search engines, misses are search results preceding the one relevant for a given user. For a chat bot, misses are answers which causes a user to chose other options suggested by the chat bot, or request other topics.

Topics of questions include personal finance. Twelve users (authors colleagues) asked the chat bot 15-20 questions each reflecting their financial situations, and stopped when they were either satisfied with an answer or dissatisfied and gave up. The same questions were sent to Google, and evaluators had to click on each search results snippet to get the document or a web page and decide on

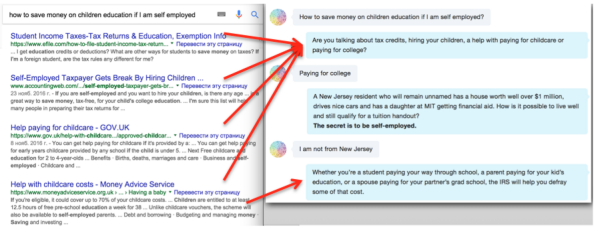


Figure 4: Comparing conventional search engine with chat bot in terms of a number of iterations

whether they can be satisfied with it.

The structure of comparison of search efficiency for the chat bot vs the search engine is shown in Fig. 4. The top portion of arrows shows that all search results (on the left) are used to form the list of topics for clarification. The arrow on the bottom shows that the bottom answer ended up being selected by the chat bot based on two rounds of user feedback and clarifications.

Parameter / search engine	**Web search**	**Chat bot**
Av. time to satisfactory search result, sec	45.3	58.1
Av. time of unsatisfactory search session (giving up and starting a new search), sec	65.2	60.5
Av. # of iter. to satisfactory search result	5.2	4.4
Av. # of iter. to unsatisfactory search result	7.2	5.6

Table 1: Comparison of the time spent and a # of iterations for the chat bot and Google search in the domain of personal finance

One can observe (Table 1) that the chat bots time of knowledge exploration session is longer than search engines. Although it might seem to be less beneficial for users, business prefer users to stay longer on their websites, as the chance of user acquisition grows. Spending 7% more time on reading chat bot answers is expected to allow a user to better familiarize himself with a domain, especially when these answers follow the selections of this user. The number of steps of an exploration session for chat bot is 25% lower than for Google search.

6 Conclusion

We conclude that using a chat bot with extended discourse tree-driven navigation is an efficient and fruitful way of information access, in comparison with conventional search engines and chat bots focused on imitation of a human intellectual activity.

The command-line demo[1] and source code[2] are available online under Apache License. Source code is a sub-project of Apache OpenNLP (`https://opennlp.apache.org/`). Since Bing search engine API is actively used for web mining to obtain candidate answers, a user would need to use her own Bing key for commercial use available at `https://datamarket.azure.com/dataset/bing/search`.

7 Acknowledgements

This work was supported by Knowledge-Trail Inc., Samur.ai, Sysomos Inc., RFBR grants #16-01-00583 and #16-29-12982 and was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'.

References

Rik Crutzen, Gjalt-Jorn Y. Peters, Sarah Dias Portugal, Erwin M. Fisser, and Jorne J. Grolleman. 2011. An artificially intelligent chat agent that answers adolescents' questions related to sex, drugs, and alcohol: an exploratory study. *Journal of Adolescent Health*, 48(5):514–519.

Boris Galitsky, Dmitry I. Ilvovsky, Sergei O. Kuznetsov, and Fedor Strok. 2013. Matching sets of parse trees for answering multi-sentence questions. In *RANLP*, pages 285–293.

Boris A. Galitsky, Dmitry Ilvovsky, Sergei O. Kuznetsov, and Fedor Strok. 2014. Finding maximal common sub-parse thickets for multi-sentence search. In *Graph Structures for Knowledge Representation and Reasoning*, pages 39–57. Springer International Publishing.

Shafiq R. Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Gwydion M. Williams. 2012. Chatbot detector. *New Scientist*, 215(2873):28.

[1] `https://1drv.ms/u/s!AlZzGY7TCKAChBqHsfVCeNZ2V_KL`

[2] `https://github.com/bgalitsky/relevance-based-on-parse-trees`

Marine Variable Linker: Exploring Relations between Changing Variables in Marine Science Literature

Erwin Marsi[1], **Pinar Øzturk**[1], **Murat V. Ardelan**[2]
Department of Computer Science[1], Department of Chemistry[2]
Norwegian University of Science and Technology
{emarsi,pinar,murat.v.ardelan}@ntnu.no

Abstract

We report on a demonstration system for text mining of literature in marine science and related disciplines. It automatically extracts variables (e.g. *CO2*) involved in events of change/increase/decrease (e.g *increasing CO2*), as well as co-occurrence and causal relations among these events (e.g. *increasing CO2 causes a decrease in pH in seawater*), resulting in a big knowledge graph. A web-based graphical user interface targeted at marine scientists facilitates searching, browsing and visualising events and their relations in an interactive way.

1 Introduction

Progress in science relies significantly on the premise that – in addition to other methods for gaining knowledge such as experiments and modelling – new knowledge can be inferred by combining existing knowledge found in the literature. Unfortunately such knowledge often remains undiscovered because individual researchers can realistically only read a relatively small part of the literature, typically mostly in the narrow field of their own expertise (Swanson, 1986). Therefore we need software to help researchers managing the ever growing scientific literature and quickly fulfil their specific information needs. Even more so for "big problems" in science, such as climate change, which require a system-level, cross-disciplinary approach.

Text mining of scientific literature has been pioneered in biomedicine and is now finding its way to other disciplines, notably in the humanities and social sciences, holding the promise for knowledge discovery from large text collections. Still, multidisciplinary fields such as marine science, climate science and environmental science remain mostly unexplored. Due to significant differences between the conceptual frameworks of biomedicine and other disciplines, simply "porting" the biomedical text mining infrastructure to another domain will not suffice. Moreover, the type of questions to be asked and the answers expected from text mining may be quite different.

Theories and models in marine science typically involve changing variables and their complex interactions, which includes correlations, causal relations and chains of positive/negative feedback loops, where multicausal events are common. Many marine scientists are thus interested in finding evidence – or counter-evidence – in the literature for events of change and their relations. Here we report on an end-user system, resulting from our ongoing work to automatically extract, relate, query and visualise events of change and their direction of variation.

Our text mining efforts in the marine science domain are guided by a basic conceptual model described in (Marsi et al., 2014). The system presented here covers a subset of this model, namely, change events, variables and causal relations. A *change* is an event in which the value of a variable is changing, but the direction of change is unspecified. There are two specific subtypes of a change event: an *increase* in which direction of change is positive and a *decrease* in which the direction of change is negative. A *variable* is something mentioned in the text that is changing its value. This is a very broad definition that covers much more than traditional entities (e.g. person, disease or protein) and includes long and complex expressions. A *cause* is relation that holds between a pair of events in which a change in one variable causes a change in another variable. An example of a sentence annotated according to this conceptual model is shown in Figure 1.

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 91–94

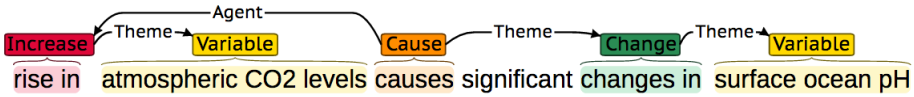


Figure 1: Example of text annotation according to conceptual model

2 Text mining system

Our text mining system for marine science literature is called *Megamouth*, inspired by the filter-feeder sharks which filter plankton out of the water. Its overall task is to turn unstructured data (text) into structured data (graph database) adhering to the conceptual model (discarding all other information) through a process of information extraction. The process is essentially a pipeline of processing steps, as briefly described below.

Step 1: Document retrieval involves crawling the websites for a predefined set of journals and extracting the text segments of interest from the HTML code, which includes title, authors, abstract, references, etc. Marine-related articles are selected through a combination of term matching with a manually compiled list of key words and a LDA topic model.

Step 2: Linguistic analysis consists of tokenisation, sentence splitting, lemmatisation, POS tagging and constituency parsing using the Stanford CoreNLP tools (Manning et al., 2014). It provides essential information required in subsequent processing such as variable extraction by pattern matching against syntactic parse trees.

Step 3: Variable and event extraction is performed simultaneously through tree pattern matching, where manually written patterns are matched against lemmatised constituency trees of sentences to extract events (increase/decrease/change) and their variables. It depends on two tools that are part of CoreNLP: Tregex is a library for matching patterns in trees based on tree relationships and regular expression matches on nodes; Tsurgeon is a closely related library for transforming trees through sequences of tree operations. For more details, see (Marsi and Øzturk, 2016; Marsi and Öztürk, 2015).

Step 4: Generalisation of variables addresses variables that are very long and complex and therefore unlikely to occur more than once. These are generalised (abstracted) by removing non-essential words and/or splitting them into atomic variables. For example, the variable *the annual, Milankovitch and continuum temperature* is split into three parts, one of which is *annual temperature*, which is ultimately itself generalised to *temperature*. This is accomplished through progressive pruning of a variable's syntactic tree, using a combination or tree pattern matching and tree operations.

Step 5: Relation extraction again uses tree-pattern matching with hand-written patterns to extract causal relations between pairs of events, identifying their cause and relation roles.

Step 6: Conversion to graph All extracted variables, events and relations are subsequently converted to a single huge property graph, which is stored and indexed in a Neo4j graph database[1] (Community Edition) to facilitate fast search and retrieval. It contains nodes for variables, generalised variables, event instances, event relations, sentences and articles. It has edges between, e.g., a variable and its generalisations. Properties on nodes/edges hold information like a sentence's number and character string on sentence nodes, or the character offsets for event instances.

Step 7: Graph post-processing enriches the initial graph in a number of ways using the Cypher graph query language. Event instance nodes are aggregated in event type nodes. Likewise, causal relation instances are aggregated in causal relations types between event types. Furthermore, co-occurrence counts for event pairs occurring in the same sentence are computed and added as co-occurrence relations between their respective event type nodes. Post-processing also includes addition of metadata and citation information, obtained through the Crossref metadata API, to articles nodes in the graph.

The final output is a big knowledge graph (millions of nodes) containing all information extracted from the input text. The graph can be searched in many different ways, depending on interest, using the Cypher graph query language. One possibility is searching for chains of causal relations. The user interface described in the next section offers a more user-friendly way of searching for a certain type of patterns, namely, relations between changing variables.

[1] https://neo4j.com/

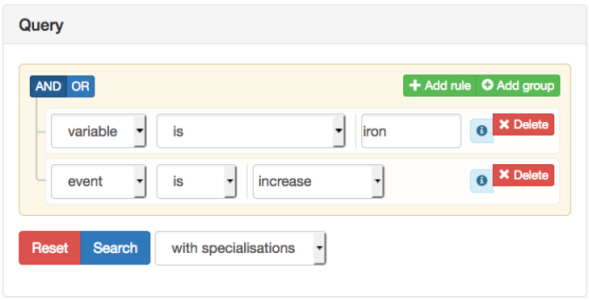


Figure 2: Example of event query composition

3 User interface

Although graph search queries can be written by hand, it takes time, effort and a considerable amount of expertise. In addition, large tables are difficult to read and navigate, lacking an easy way to browse the results, e.g., to look up the source sentences and articles for extracted events. Moreover, users need to have a local installation of all required software and data. The Marine Variable Linker (MVL) is intended to solve these problems. Its main function is to enable non-expert users (marine scientists) to easily search the graph database in an interactive way and to present search results in a browsable and visual way. It is a web application that runs on any modern platform with a browser (Linux, Mac OS, Windows, Android, iOS, etc). It is a graphical user interface, which relies on familiar input components such as buttons and selection lists to compose queries, and uses interactive tables and graphs to present search results. Hyperlinks are used for navigation and to connect related information, e.g. the webpage of the source journal article.

Figure 2 shows an example of a search query for events consisting of two search rules: (1) the variable equals *iron* and (2) the event type is *increase*. In addition, the search is *with specialisations*, which means that it includes variables that can be generalised to *iron*, such as *particulate iron* or *iron in clam mactra lilacea*. More rules can be added to narrow down, or widen, event search. Clicking the search button will bring up a new table for matching event types, showing the instance counts, predicates and variables (not shown here). Clicking on any row in this event types table will bring up the corresponding event instances table, which shows all the actual mentions of this event in journal articles. Each row in the instance table shows a sentence, year of publication and source.

Once events are defined, one can search for relations between these events, where queries can be composed in a similar fashion as for events. The first kind of relation is *cooccurs*, which means that two events co-occur in the same sentence. When two events are frequently found together in a single sentence, they tend to be associated in some way, possibly by correlation. The second kind of relation is *causes*, which means that two events in a sentence are causally related, where one event is the cause and other the effect. Causality must be explicitly described in the sentence, for example, by words such as *causes*, *therefore*, *leads to*, etc.

Relation search results are presented in two ways. The relation types table contains all pairs of event types, specifying their relation, event predicates, event variables and counts. Figure 3 provides an example for an open-ended search query where the cause is an event with variable *iron* (including its specialisations, direction of change unspecified), whereas the effect is left open (i.e., can be any event). The corresponding relation graph is shown in Figure 4. The nodes are event types with red triangles for increases, blue triangles for decreases and green diamonds for changes.

Clicking on a row in the table or a node in the graph brings up a corresponding instances table (cf. bottom of Figure 3), showing sentences, years and citations of articles containing the given relation. The events are marked in colour: red for increasing, blue for decreasing and green for changing. Hovering the mouse over the document icon will show a citation for the source article, whereas clicking it will open the article's web page containing the sentence in a new window.

A demo of an MVL instance indexing 75,221 marine-related abstracts from over 30 journals is currently freely accessible on the web.[2] Source code for the text mining system and the graphical user interface is freely available.[3]

4 Discussion and Future work

Our system still makes many errors. Variables and events are sometimes incorrectly extracted (e.g. variable *more iron* in Figure 3 ought to be just *iron*), often due to syntactic parsing errors, and many are missed altogether (e.g. the decline in *the particulate organic carbon quotas* in the same Figure), because events can be expressed in so

[2] http://baleen.idi.ntnu.no/demos/megamouth-abs/

[3] https://github.com/OC-NTNU

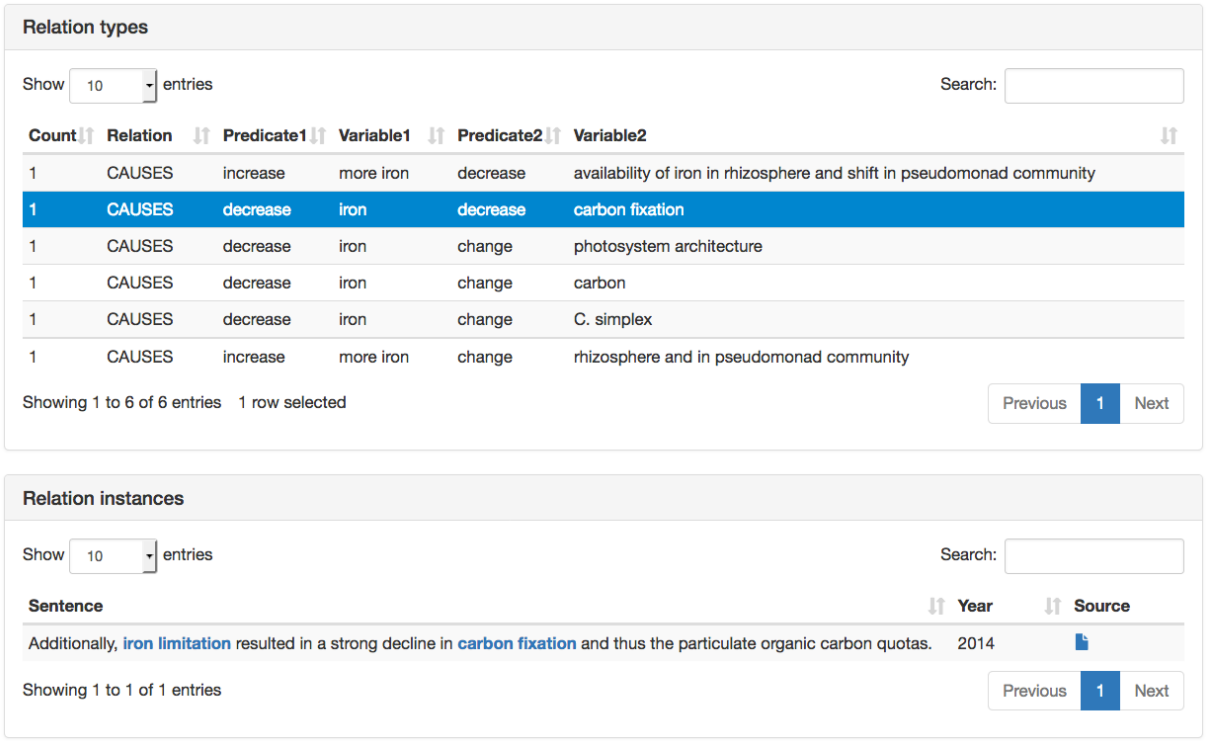


Figure 3: Example of search results for causal relations types (top) and a selected instance (bottom)

many different and complex ways. Yet we feel that even with a fair amount of noise, the current proof-of-concept already offers practical merit in battling the literature deluge. We will continue to work on improving recall and precision, as well as usability aspects of the interface. One aspect under active development is the integration of new and better algorithms for causal relation extraction based on machine learning from manually annotated data. The knowledge graph can be searched efficiently using Cypher queries, which opens up many other interesting opportunities for knowledge discovery. We hope our system will attract interest from marine and climate scientists, raising awareness of the potential of text mining, as progress will crucially depend on building a community similar to that in biomedical text mining.

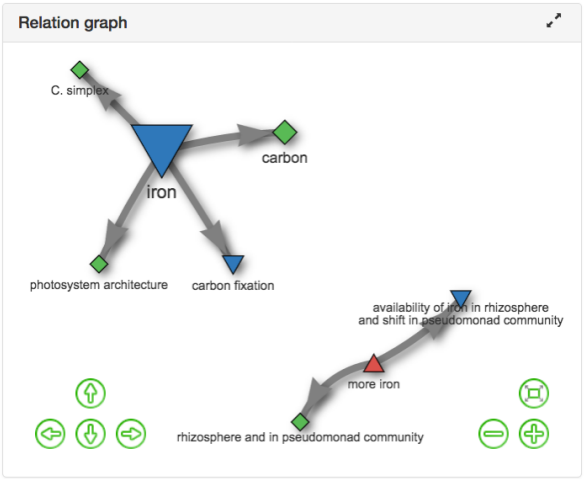


Figure 4: Example of relation graph

Acknowledgments

Financial aid from the European Commission (OCEAN-CERTAIN, FP7-ENV-2013-6.1-1; no:603773) is gratefully acknowledged.

References

C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

E. Marsi and P. Öztürk. 2015. Extraction and generalisation of variables from scientific publications. In *EMNLP*, pages 505–511, Lisbon, Portugal.

E. Marsi and P. Øzturk. 2016. Text mining of related events from natural science literature. In *Workshop on Semantics, Analytics, Visualisation: Enhancing Scholarly Data (SAVE-SD)*, Montreal, Canada.

E. Marsi, P. Öztürk, E. Aamot, G. Sizov, and M.V. Ardelan. 2014. Towards text mining in climate science: Extraction of quantitative variables and their relations. In *Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing*, Reykjavik, Iceland.

D. R. Swanson. 1986. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18.

Neoveille, a Web Platform for Neologism Tracking

Emmanuel Cartier
Université Paris 13 Sorbonne Paris Cité - LIPN - RCLN UMR 7030 CNRS
99 boulevard Jean-Baptiste Clément
93430 Villetaneuse, FRANCE
emmanuel.cartier@lipn.univ-paris13.fr

Abstract

This paper details a software designed to track neologisms in seven languages through newspapers monitor corpora. The platform combines state-of-the-art processes to track linguistic changes and a web platform for linguists to create and manage their corpora, accept or reject automatically identified neologisms, describe linguistically the accepted neologisms and follow their lifecycle on the monitor corpora. In the following, after a short state-of-the-art in Neologism Retrieval, Analysis and Life-tracking, we describe the overall architecture of the system. The platform can be freely browsed at www.neoveille.org where detailed presentation is given. Access to the editing modules is available upon request.

1 Credits

Neoveille is an international Project funded by the ANR IDEX specific funding scheme. It gathers seven Linguistics and Research Centers. See website for details.

2 Introduction

Linguistic change is one of the fundamental properties of language, even if, at least on a short-term period, languages appear to be extremely conservative and reluctant to change. Whereas NLP efforts have mainly focused on synchronic language analysis, research and applications are very sparse on the diachronic side, especially concerning short term diachrony. But, with the availability of big web corpora, the maturity of automatic linguistic analysis and especially those able to process big data while maintaining a reasonable quality, it is now possible to monitor language change and track linguistic innovations.

3 Previous Work in Neology and Neology Tracking

Linguistic change has been studied for decades and even centuries in linguistics, and has been dealt with more recently in computational linguistics.

3.1 Linguistic Neology Models

3.1.1 Neologism Categories

Linguistic change has been first focused on by the Comparative Grammars School, whose main goal was to study languages diachronically. They have mainly based their descriptions and analysis on linguistic forms, describing phonetical, phonological, morphological, syntactical and semantic change on a long-term basis (Geeraerts, 2010). More recently, several attempts have emerged in the field of linguistic change, mainly focusing on the lexical units and proposing typology of neologisms (Schmid, 2015),(Sablayrolles, 2016).

3.1.2 Synchrony, Diachrony, Diastraty

A complementary approach is due to (Gevaudan and Koch, 2010) who state that every lexical evolution can be described through three parameters : two are universal, the semantic parameter (explicating a continuity of meaning or a discontinuity, in this case further described) and the stratic parameter (linking the linguistic structure to its sociological context : borrowings are explained this way); the third one is linked to every specific linguistic formal structure, with four generic matrices : conversion, morphological extension, composition and clipping).

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 95–98

3.1.3 Neologism Life-cycle(s)

Neology is one of the aspect of linguistic change, with necrology and stability. One important aspect is thus to model the lifecycle of a neologism, from its first occurrence to its potential disappearance or conventionalization. (Traugott and Trousdale, 2013) have proposed three salient states : innovation, propagation and conventionalisation, linking each to several more-or-less obvious properties. With these models in mind, NLP has developed several algorithms to track and study the lifecycle of neologisms.

3.2 Computational Models of Neology

Computational Linguistics has begun to work on linguistic change not long ago, mainly because it needs to have at hand large diachronic electronic corpora. Neology is moreover still considered as an secondary topic, as novel lexical units represents less than 5 percent of lexical units in corpora, according to several studies. But linguistic change is the complementary aspect of the synchronic structure. Every lexical unit is subjected to time, form and meaning can change, due to diastratic events and situations. The advent of electronic (long and short-term) diachronic corpora, scientific research and advances on word-formation and machine learning techniques able to manage big corpora, have permitted the emergence of neology tracking systems. Apart from a best knowledge of language lifecyle(s), these tools would permit to update lexicographic resources, computational assets and parsers.

From the CL point of view, the main questions are : how can we automatically track neologisms, categorize them and follow their evolution, from their first appearance to their conventionalisation or disappearance? At best, can we induce neology-formation procedures from examples and therefore predict potential neologisms?

3.3 Existing Neology Tracking System

3.3.1 the Exclusion Dictionary Architecture (EDA)

The main achievement of neology tracking consists in system extracting novel forms from monitor corpora, using lexicographic resources as a reference exclusion dictionary to induce unknown words, what we can call the ”exclusion dictionary architecture” (EDA). The first system is due to (Renouf, 1993) for English : a monitor corpora and a reference dictionary from which unknown words can be derived. Further filters then apply to eliminate spellings errors and Proper Nouns. Subsequent developments all replicate this architecture : OBNEO (Cabré and De Yzaguirre, 1995), NeoCrawler (Kerremans et al., 2012), Logoscope (Gérard et al., 2014) and more recently Neoveille (Cartier, 2016).

Four main difficulties arise from these architecture : first, EDA can not track semantic neologisms, as they use existing lexical units to convey innovative meanings; second, the design of a reference exclusion dictionary is not that obvious as it requires the existence of a machine-readable dictionary : this entails specific procedures to apply this architecture to less-resourced languages, and the availability of an up-to-date machine-readable dictionary for more resourced languages ; third, the EDA architecture is not sufficient in itself : among unknown forms, most of them are Proper Nouns, spelling mistakes and other cases derived from corpus boilerplate removal : this entails a post-processing phase to depart cases; Fourth, these systems do not take into account the sociological and diatopic aspects of neologism, as they limit their corpora to specific domains : a ideal system should be able to extend its monitoring to new corpora and maintain diastratic meta-datas to characterize novel forms. To the best of our knowledge, Neoveille (Cartier, 2016) is the only system implementing this aspect.

3.3.2 Semantic Neology Approaches

As for semantic neology, three approaches have been recently proposed, none of them being exploited in an operational system. The first one stems from the idea that meaning change is linked to domain change : every texts and thus the constituent existing lexical units are assigned one or more topic; if a lexical unit emerges in a new domain, a change in meaning should have occurred (Gérard et al., 2014). The main drawback of this approach is that it is limited to specific semantic change (it can not tackle conventional metaphors if appeared in the same domain, nor detect extension or restriction of meaning) and mainly limited to Nouns.

An other approach is linked to the distributional paradigm : ”You shall know a word by the company it keeps”(Firth, 1957). The main idea is to retrieve from a large corpora all the collocates or collostructions, and classify them according to sev-

eral metrics. The main salient resulting context words represent the "profile" (Blumenthal, 2009) or "sketch" (Kilgarriff et al., 2004) of a lexical unit for the given synchronic period. The most elaborated system is surely the Sketch Engine system, which propose for every lexical engine its "sketch", i.e. a list, for any user-defined syntactic schemas (for example modifiers, nominal subject, object and indirect object for verb) of occurrences, sorted by one or several association measure. This system can be improved in two main ways : first, it does not propose complete syntactic schemas for lexical units like verbs (it is limited to either a SUBJ-VERB or VERB-OBJ relation, but does not propose SUBJ-VERB-OBJ relations); second, it does not propose a clustering of occurrences, whereas distributional semantics could fill the gap and propose distributional classes at any place in the schema, instead of flat list of occurrences.

A third approach consists in tracking semantic change by applying the second aspect of the distributional hypothesis, that lexical units sharing the most of contexts are most likely to be semantically similar. This assumption has been applied to many computational semantic tasks. Applied to semantic change, if you have at your disposal a bunch of diachronic corpora, you can build the semantic vectors of any lexical unit corresponding to several periods, and track the changes from one period to another. First experiments have been proposed by (Hamilton et al., 2016). The main advantage of this approach resides in the fact that it proposes for a given word a list of semantically similar words, among which synonyms and hypernyms, which permits to clearly explicit the meaning of a word. The main drawback of this approach is to be unable to distinguish meanings for polysemous units. Another relative drawback relies on the fuzzy notion of similarity, which results in semantically too-slightly similar words (analogy), or even opposite words (antonymy). But this approach is clearly of great help to humanly grasp the meaning of a word.

In the Neoveille Project, we are currently developing a approach combining the Sketch approach mixed with semantic distributions on the main lexical unit and its arguments.

3.3.3 Tracking the Lifecycle of Neologisms

In our view, we postulate that neologisms are new form-meaning pairs (Lexical units) and thus exist from their first occurrence. Tracking the lifecycle of neologisms requires to fix criteria to identify the main phases : emergence, dissemination, conventionalization (Traugott and Trousdale, 2013). In operational systems, the main tool to follow the life of a neologism is the timeline rendering the absolute or relative frequency of the lexical unit. In Neoveille, these figures are relative to specific diastratic and diatopic paramaters, visually enabling to distinguish emergence, spread and conventionalization. These analysis are available for each identified neologism by clicking on the stats icon (see website, last neologisms menu).

4 Neoveille Tracking System Architecture

The Neoveille architecture aims at enabling a complete synergy between NLP system and expert linguists : expert linguists are not able to monitor the vast amount of textual data whereas automatic processes can help tackle this amount: experts can accurately decide if a word is or is not a neologism; our current point of view is that linguists must have the last word on what is and is not a neologism, and on the linguistic description; but as knowledge and description will grow up with time, we will build Supervised Machine Learning techniques able to predict potential neologisms.

The Neoveille web architecture has five main components:

1. A corpora manager: corpora is the main feed for NLP systems, and we propose to linguists a system enabling to choose their corpora and to add to them several meta-datas. The corpora, once defined by the user, are retrieved on a daily basis, indexed and searched for neologisms. Corpora management is available in the restricted area on the left menu;

2. An advanced search engine on the corpora; : not only corpora can be monitored, but also the system should propose a search engine with advanced capabilities : advanced querying, filtering and faceting of results; the Neoveille search engine is available on the restricted area on the left menu; based on Apache Solr, it enables to query the corpora in a multifactorial manner, with facets and visual filters;

3. Advanced Data Analytics expliciting the lifecycle of neologisms and their diachronic, di-

atopic and diastratic parameters : Neoveille provides such a Data Analytics Framework by combining meta-data to text mining; These visual analysis are available for every neologisms by clicking the stats icon;

4. A linguistic description component for neologisms : this module, whose microstructure has been setup for several years, could be used for knowledge of neology in a given language, and could also be used by a supervised machine learning system, as these features include a lot of formal properties. This component is accessible in the restricted area on the left menu.

5. formal and semantic neologisms tracking with state-of-the-art techniques The formal and semantic neologism components are accessible in the restricted area on the left menu. They work for the seven languages of the project.

5 Conclusion and perspectives

This short presentation has evoked the design and the overall architecture of a software for linguistic analysis focusing on linguistic change in a contemporary monitor corpora. It has several interesting properties :

- real-time tracking, analysis and visualization of linguistic change;
- complete synergy between Computational Linguistics processing and linguistic experts intuitions and knowledge, especially the possibility of editing automatic results by experts and exploitation of linguistic annotations by machine learning processes;
- modularity of software : corpora management, state-of-the-art search engine including analysis and visualization, neologisms mining, neologism linguistic description, lifecycle tracking.

This project, currently focusing on seven languages is in the path to extend to other languages.

References

P. Blumenthal. 2009. Éléments d'une théorie de la combinatoire des noms. *Cahiers de lexicologie 94 (2009-1), 11-29.*

Maria Teresa Cabré and Luis De Yzaguirre. 1995. Stratégie pour la détection semi-automatique des néologismes de presse. *TTR : traduction, terminologie, rédaction, 8 (2), p. 89-100.*

Emmanuel Cartier. 2016. Néoveille, système de repérage et de suivi des néologismes en sept langues. *Neologica, 10, Revue internationale de néologie, p.101-131.*

John R. Firth. 1957. Papers in linguistics 1934–1951, london, oxford university press, 1957.

Dirk Geeraerts. 2010. *Theories of Lexical Semantics*. Oxford University Press.

Paul Gevaudan and Peter Koch. 2010. Sémantique cognitive et changement sémantique. *Grandes voies et chemins de traverse de la sémantique cognitive, Mémoire de la Société de linguistique de Paris, XVIII, pp. 103-145.*

Christophe Gérard, Ingrid Falk, and Delphine Bernhard. 2014. Traitement automatisé de la néologie : pourquoi et comment intégrer l'analyse thématique? *Actes du 4e Congrès mondial de linguistique française (CMLF 2014), Berlin, p. 2627-2646.*

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. *ACL 2016.*

Daphné Kerremans, Susanne Stegmayr, and Hans-Jörg Schmid. 2012. The NeoCrawler: identifying and retrieving neologisms from the internet and monitoring on-going change. *Kathryn Allan and Justyna A. Robinson, eds., Current methods in historical semantics, Berlin etc.: de Gruyter Mouton, 59-96.*

A. Kilgarriff, R. Pavel, Pavel S., and Tugwell D. 2004. The Sketch Engine. *Proceedings of Euralex, pages 105–116, Lorient.*

Antoinette Renouf. 1993. Sticking to the Text : a corpus linguist's view of language. *ASLIB Proceedings, 45 (5), p. 131-136.*

Jean-François Sablayrolles. 2016. *Les néologismes*. Collection Que sais-je? Presses Universitaires de France.

Hans-Jörg Schmid. 2015. The scope of word-formation research. *Peter O. Müller, Ingeborg Ohnheiser, Susan Olsen and Franz Rainer, eds., Word-Formation. An International Handbook of the Languages of Europe. Vol. I.*

Elizabeth Closs Traugott and Graeme Trousdale. 2013. *Constructionalization and constructional changes.*

Building Web-Interfaces for Vector Semantic Models with the WebVectors Toolkit

Andrey Kutuzov
University of Oslo
Oslo, Norway
andreku@ifi.uio.no

Elizaveta Kuzmenko
Higher School of Economics
Moscow, Russia
eakuzmenko_2@edu.hse.ru

Abstract

We present WebVectors, a toolkit that facilitates using distributional semantic models in everyday research. Our toolkit has two main features: it allows to build web interfaces to query models using a web browser, and it provides the API to query models automatically. Our system is easy to use and can be tuned according to individual demands. This software can be of use to those who need to work with vector semantic models but do not want to develop their own interfaces, or to those who need to deliver their trained models to a large audience. WebVectors features visualizations for various kinds of semantic queries. For the present moment, the web services with Russian, English and Norwegian models are available, built using WebVectors.

1 Introduction

In this demo we present *WebVectors*, a free and open-source toolkit[1] helping to deploy web services which demonstrate and visualize distributional semantic models (widely known as word embeddings). We show its abilities on the example of the living web service featuring distributional models for English and Norwegian[2].

Vector space models, popular in the field of distributional semantics, have recently become a buzzword in natural language processing. In fact, they were known for decades, and an extensive review of their development can be found in (Turney et al., 2010). Their increased popularity is mostly due to the new prediction-based approaches, which allowed to train distributional models with large amounts of raw linguistic data very fast. The most established word embedding algorithms in the field are highly efficient *Continuous Skip-Gram* and *Continuous Bag-of-Words*, implemented in the famous *word2vec* tool (Mikolov et al., 2013b; Baroni et al., 2014), and *GloVe* introduced in (Pennington et al., 2014).

Word embeddings represent the meaning of words with dense real-valued vectors derived from word co-occurrences in large text corpora. They can be of use in almost any linguistic task: named entity recognition (Siencnik, 2015), sentiment analysis (Maas et al., 2011), machine translation (Zou et al., 2013; Mikolov et al., 2013a), corpora comparison (Kutuzov and Kuzmenko, 2015), word sense frequency estimation for lexicographers (Iomdin et al., 2016), etc.

Unfortunately, the learning curve to master word embedding methods and how to present the results to general public may be steep, especially for people in (digital) humanities. Thus, it is important to facilitate research in this field and to provide access to relevant tools for various linguistic communities.

With this in mind, we are developing the *WebVectors* toolkit. It allows to quickly deploy a stable and robust web service for operations on word embedding models, including querying, visualization and comparison, all available even to users who are not computer-savvy.

WebVectors can be useful in a very common situation when one has trained a distributional semantics model for one's particular corpus or language (tools for this are now widespread and simple to use), but then there is a need to demonstrate the results to the general public. The toolkit can be installed on any Linux server with a small set of standard tools as prerequisites, and generally works out-of-the-box. The administrator needs

[1] https://github.com/akutuzov/webvectors

[2] http://ltr.uio.no/semvec

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 99–103

only to supply a trained model or models for one's particular language or research goal. The toolkit can be easily adapted for specific needs.

2 Deployment

The toolkit serves as a web interface between distributional semantic models and users. Under the hood it uses the following software:

- *Gensim* library (Řehůřek and Sojka, 2010) which is responsible for actual interaction with models[3];
- Python *Flask* framework responsible for the user interface. It runs either on top of a regular *Apache* HTTP server or as a standalone service (using *Gunicorn* or other standalone WSGI server).

Flask communicates with *Gensim* (functioning as a daemon with our wrapper) via sockets, sending user queries and receiving answers from models.

This architecture allows fast simultaneous processing of multiple users querying multiple models over network. Models themselves are permanently stored in memory, eliminating time-consuming stage of loading them from permanent storage every time there is a need to process a query.

The setup process is extensively covered by the installation instructions available at `https://github.com/akutuzov/webvectors`.

3 Main features of WebVectors

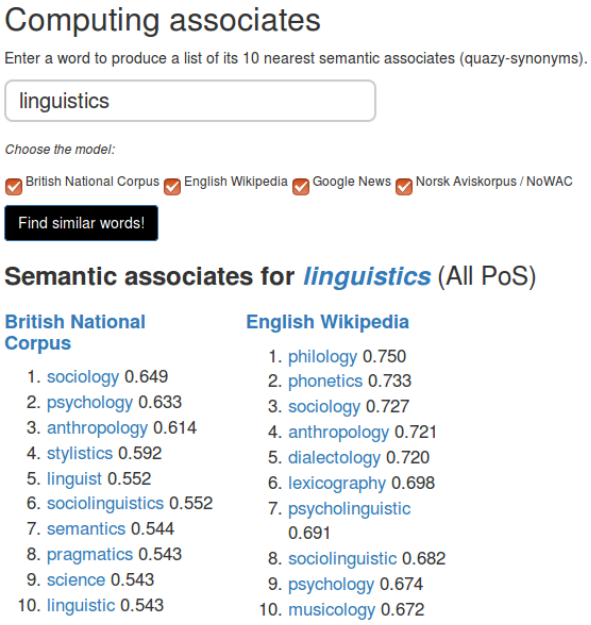
Once *WebVectors* is installed, one can interact with the loaded model(s) via a web browser. Users are able to:

1. find **semantic associates**: words semantically closest to the query word (results are returned as lists of words with corresponding similarity values); an illustration of how it looks like in our demo web service is in Figure 1;

2. calculate exact **semantic similarity** between pairs of words (results are returned as cosine similarity values, in the range between -1 and 1);

[3]Can be any distributional model represented as a list of vectors for words: *word2vec*, *GloVe*, etc.

Figure 1: Computing associates for the word *'linguistics'* based on the models trained on English Wikipedia and BNC.



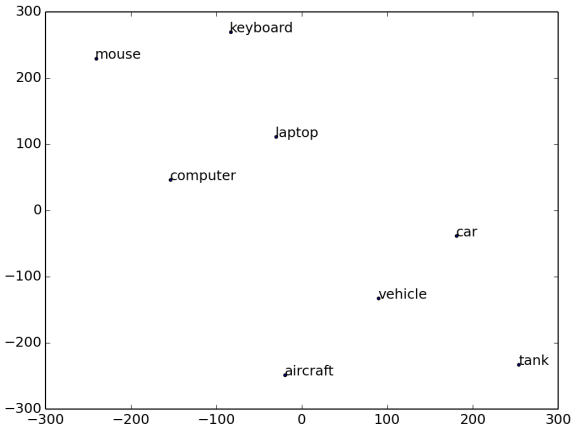
3. apply **algebraic operations** to word vectors: addition, subtraction, finding average vector for a group of words (results are returned as lists of words nearest to the product of the operation and their corresponding similarity values); this can be used for analogical inference, widely known as one of the most interesting features of word embeddings (Mikolov et al., 2013b);

4. **visualize** semantic relations between words. As a user enters a set of words, the service builds a map of their inter-relations in the chosen model, and then returns a 2-dimensional version of this map, projected from the high-dimensional vector space, using *t-SNE* (Van der Maaten and Hinton, 2008). An example of such visualization is shown in Figure 2;

5. get the **raw vector** (array of real values) for the query word.

One can use part-of-speech filters in all of these operations. It is important to note that this is possible only if a model was trained on a PoS-tagged corpus and the tags were added to the resulting lemmas or tokens. Obviously, the tagger should

Figure 2: Visualizing the positions of several words in the semantic space



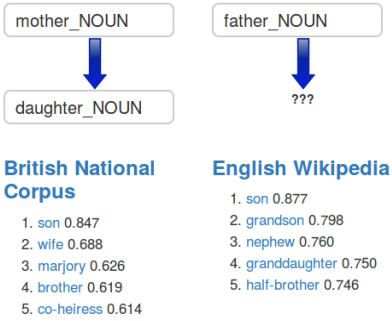
differentiate between homonyms belonging to different parts of speech. *WebVectors* can also use an external tagger to detect PoS for query words, if not stated explicitly by the user. By default, *Stanford CoreNLP* (Manning et al., 2014) is used for morphological processing and lemmatization, but one can easily adapt *WebVectors* to any other PoS-tagger.

In fact, one can use not only PoS tags, but any other set of labels relevant for a particular research project: time stamps, style markers, etc. *WebVectors* will provide the users with the possibility to filter the models' output with respect to these tags.

Another feature of the toolkit is the possibility to display results from more than one model simultaneously. If several models are enumerated in the configuration file, the *WebVectors* daemon loads all of them. At the same time, the user interface allows to choose one of the featured models or several at once. The results (for example, lists of nearest semantic associates) for different models are then presented to the user side-by-side, as in the Figure 3. This can be convenient for research related to comparing several distributional semantic models (trained on different corpora or with different hyperparameters).

Last but not least, *WebVectors* features a simple API that allows to query the service automatically. It is possible to get the list of semantic associates for a given word in a given model or to compute semantic similarity for a word pair. The user performs GET requests to URLs following a

Figure 3: Analogical inference with several models



specific pattern described in the documentation; in response, a file with the first 10 associates or the semantic similarity score is returned. There are two formats available at the present moment: *json* and tab-separated text files.

4 Live demos

The reference web service running on our code base is at `http://ltr.uio.no/semvec`. It allows queries to 4 English models trained with the *Continuous Skipgram* algorithm (Mikolov et al., 2013b): the widely known Google News model published together with the *word2vec* tool, and the models we trained on Gigaword, British National Corpus (BNC) and English Wikipedia dump from September 2016 (we plan to regularly update this last one). Additionally, it features a model trained on the corpus of Norwegian news texts, *Norsk aviskorpus* (Hofland, 2000). To our knowledge, this is the first neural embedding model trained on the Norwegian news corpus made available online; (Al-Rfou et al., 2013) published distributional models for Norwegian, but they were trained on the Wikipedia only, and did not use the current state-of-the-art algorithms.

Prior to training, each word token in the training corpora was not only lemmatized, but also augmented with a Universal PoS tag (Petrov et al., 2012) (for example, *boot_VERB*). Also, some amount of strongly related bigram collocations like '*Saudi::Arabia_PROPN*' was extracted, so that they receive their own embeddings after the training. The Google News model already features ngrams, but lacks PoS tags. To make it more comparable with other models, we assigned each word in this model a PoS tag with *Stanford CoreNLP*.

Another running installation of our toolkit is the *RusVectores* service available at `http://`

rusvectores.org (Kutuzov and Kuzmenko, 2016). It features 4 *Continuous Skipgram* and *Continuous Bag-of-Words* models for Russian trained on different corpora: the Russian National Corpus (RNC)[4], the RNC concatenated with the Russian Wikipedia dump from November 2016, the corpus of 9 million random Russian web pages collected in 2015, and the Russian news corpus (spanning time period from September 2013 to November 2016). The corpora were linguistically pre-processed in the same way, lending the models the ability to better handle rich morphology of Russian. The *RusVectores* is already being employed in academic studies in computational linguistics and digital humanities (Kutuzov and Andreev, 2015; Kirillov and Krizhanovskij, 2016; Loukachevitch and Alekseev, 2016) (several other research projects are in progress as of now).

One can use the aforementioned services as live demos to evaluate the *WebVectors* toolkit before actually employing it in one's own workflow.

5 Conclusion

The main aim of *WebVectors* is to quickly deploy web services processing queries to word embedding models, independently of the nature of the underlying training corpora. It allows to make complex linguistic resources available to wide audience in almost no time. We continue to add new features aiming at better understanding of embedding models, including sentence similarities, text classification and analysis of correlations between different models for different languages. We also plan to add models trained using other algorithms, like *GloVe* (Pennington et al., 2014) and *fastText* (Bojanowski et al., 2016).

We believe that the presented open source toolkit and the live demos can popularize distributional semantics and computational linguistics among general public. Services based on it can also promote interest among present and future students and help to make the field more compelling and attractive.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1.

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

Knut Hofland. 2000. A self-expanding corpus based on newspapers on the web. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*.

B. L. Iomdin, A. A. Lopukhina, K. A. Lopukhin, and G. V. Nosyrev. 2016. Word sense frequency of similar polysemous words in different languages. In *Computational Linguistics and Intellectual Technologies. Dialogue*, pages 214–225.

A. N. Kirillov and A. A. Krizhanovskij. 2016. Model' geometricheskoj struktury sinseta [the model of geometrical structure of a synset]. *Trudy Karel'skogo nauchnogo centra Rossijskoj akademii nauk*, (8).

Andrey Kutuzov and Igor Andreev. 2015. Texts in, meaning out: neural language models in semantic similarity task for Russian. In *Computational Linguistics and Intellectual Technologies. Dialogue*, Moscow. RGGU.

Andrey Kutuzov and Elizaveta Kuzmenko. 2015. Comparing neural lexical models of a classic national corpus and a web corpus: The case for Russian. *Lecture Notes in Computer Science*, 9041:47–58.

Andrey Kutuzov and Elizaveta Kuzmenko. 2016. Webvectors: a toolkit for building web interfaces for vector semantic models (in press).

Natalia Loukachevitch and Aleksei Alekseev. 2016. Gathering information about word similarity from neighbor sentences. In *International Conference on Text, Speech, and Dialogue*, pages 134–141. Springer.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[4] http://www.ruscorpora.ru/en/

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

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

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.

Scharolta Katharina Sienčnik. 2015. Adapting word2vec to named entity recognition. In *Nordic Conference of Computational Linguistics NODALIDA 2015*, page 239.

Peter D. Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

InToEventS: An Interactive Toolkit for Discovering and Building Event Schemas

Germán Ferrero
Universidad Nacional de Córdoba
ferrero.gf@gmail.com

Audi Primadhanty
Universitat Politècnica de Catalunya
primadhanty@cs.upc.edu

Ariadna Quattoni
Xerox Research Centre Europe
ariadna.quattoni@xrce.xerox.com

Abstract

Event Schema Induction is the task of learning a representation of events (e.g., bombing) and the roles involved in them (e.g, victim and perpetrator). This paper presents InToEventS, an interactive tool for learning these schemas. InToEventS allows users to explore a corpus and discover which kind of events are present. We show how users can create useful event schemas using two interactive clustering steps.

1 Introduction

An event schema is a structured representation of an event, it defines a set of atomic predicates or facts and a set of role slots that correspond to the typical entities that participate in the event. For example, a bombing event schema could consist of atomic predicates (e.g., *detonate, blow up, plant, explode, defuse* and *destroy*) and role slots for a perpetrator (the person who detonates plants or blows up), instrument (the object that is planted, detonated or defused) and a target (the object that is destroyed or blown up). Event schema induction is the task of inducing event schemas from a textual corpus. Once the event schemas are defined, slot filling is the task of extracting the instances of the events and their corresponding participants from a document.

In contrast with information extraction systems that are based on atomic relations, event schemas allow for a richer representation of the semantics of a particular domain. But, while there has been a significant amount of work in relation discovery, the task of unsupervised event schema induction has received less attention. Some unsupervised approaches have been proposed (Chambers and Jurafsky, 2011; Cheung et al., 2013; Chambers, 2013; Nguyen et al., 2015). However, they all end up assuming some form of supervision at document level, and the task of inducing event schemas in a scenario where there is no annotated data is still an open problem. This is probably because without some form of supervision we do not even have a clear way of evaluating the quality of the induced event schemas.

Figure 1: Event Schema Definition

In this paper we take a different approach. We argue that there is a need for an interactive event schema induction system. The tool we present enables users to explore a corpus while discovering and defining the set of event schemas that best describes the domain. We believe that such a tool addresses a realistic scenario in which the user does not know in advance the event schemas that he is interested in and he needs to explore the corpus to better define his information needs.

The main contribution of our work is to present an interactive event schema induction system that can be used by non-experts to explore a corpus and

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 104–107

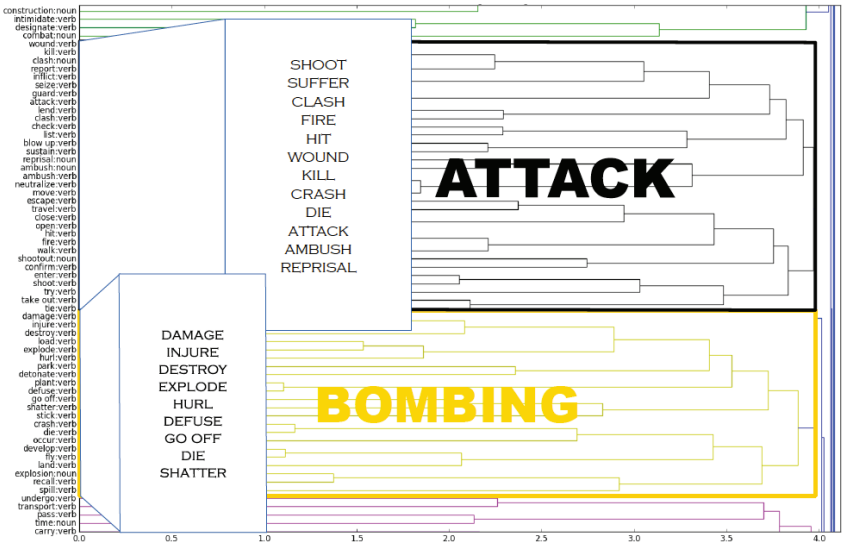


Figure 2: Event Clustering

easily build event schemas and their corresponding extractors.

The paper is organized as follows. Section 2 describes our event schema representation. Section 3 describes the interactive process for event schema discovery. Section 4 gives a short overview of related work. Finally section 5 concludes.

2 Event Schema

Our definition of event schema follows closely that of Chambers and Jurafsky (2011). An event schema consists of two main components:

Event Triggers. These correspond to a set of atomic predicates associated with an event. For example, as shown in Figure 1, for the *bombing* event schema we might have the atomic predicates: *explode, hurl, destroy, hit, shatter* and *injure*. Each atomic predicate is represented by a tuple composed of a literal (e.g. *explode*), a real-valued word vector representation and a distance threshold that defines a ball around the literal in a word vector space representation.

Event Slots. These correspond to the set of participating entities involved in the event. For example, for the *bombing* event schema we might have the event slots: *victim, perpetrator, instrument* and *target*. Each slot is represented by a tuple consisting of an entity type (e.g. person, organization or object) and a set of predicates, for example for the *victim* slot, the predicates are *injured, dies, wounded, fired* and *killed*. Each predicate is in turn represented by a tuple, consisting of a literal, a syntactic relation, a word vector and a distance threshold that defines a semantic ball. For example, the *injured* predicate is represented by the literal: *injure*, and the syntactic relation: object. This tuple is designed to represent the fact that a victim is a person or organization who has been injured, and whose corresponding word vector representation is inside a given semantic ball.

3 Event Schema Induction

We now describe InToEventS, an interactive system that allows a user to explore a corpus and build event schemas. Like in (Chambers and Jurafsky, 2011) the process is divided in two main steps. First, the user will discover the events present in the corpus (that is, event trigger sets) by interactively defining a soft partition of the predicate literals observed in the corpus. Depending on his information needs he will chose a subset of the clusters that correspond to the events that he is interested in representing. In the second step, for each chosen event trigger set, the user will complete the event schema and build slots or semantic roles via an interactive clustering of the syntactic arguments of the atomic predicates in the event trigger set.

3.1 First Step: Event Induction

To build event trigger sets we will cluster predicate literals observed in the corpus. To do this we first need to compute a distance between the predicate literals. Our notion of distance is based on two simple observations: (1) literals that tend to appear nearby in a document usually play a role in the same event description (e.g., This morning

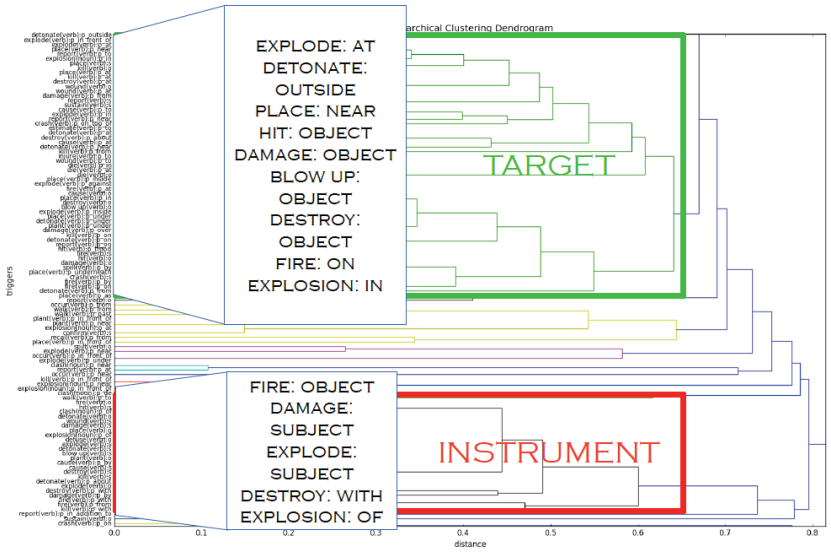


Figure 3: Slots Clustering

a terrorist *planted* a bomb, thankfully the police *defused* it before it *blew up*); and (2) literals with similar meaning are usually describing the same atomic predicates (e.g., *destroy* and *blast*).

We first extract all unique *verbs* and all nouns *noun* with a corresponding synset in Wordnet labeled as *noun.event* or *noun.act*, these are our predicate literals. Then, for each pair of literals we compute their distance taking into account the probability that they will appear nearby in a document sampled from the corpus and the distance of the corresponding literals in a word embedding vector space [1].

Once the distance is computed, we run agglomerative clustering. In the interactive step we let the user explore the resulting dendrogram and chose a distance threshold that will result in an initial partition of the predicate literals into event trigger sets (see Figure 2). In a second step, the user can merge or split the initial clusters. In a third step, the user selects and labels the clusters that he is interested in, this will become incomplete event schemas. Finally, using the word embedding representation the user has the option of expanding each event trigger set by adding predicate literals that are close in the word vector space.

3.2 Second Step: Role Induction

In this step we will complete the event schemas of the previous step with corresponding semantic roles or slots. This process is based on a simple idea: let's assume for instance a bombing event with triggers: {attack, blow up, set off, injure, die, ...}, we can intuitively describe a *victim* of a *bombing* event as *"Someone who dies, is attacked or injured"*, that is: *"PERSON: subject of die, object of attack, object of injured"*.

Recall that a slot is a set of predicates represented with a tuple composed by: a literal predicate and a syntactic relation, e.g. kill-subject. Additionaly each slot has an entity-type. In a first step, for each predicate in the event trigger set we extract from the corpus all unique tuples of the form predicate-syntactic relation-entity type. The extraction of such tuples uses the universal dependency representation computed by Stanford CoreNLP parser and named entity classifier.

In a second step, we compute a distance between each tuple that is based on the average word embeddings of the arguments observed in the corpus for a given tuple. For example, to compute a vector *(die, subject, PERSON)* we identify all entities of type *PERSON* in the corpus that are *subject* of the verb *die* and average their word embeddings.

Finally, as we did for event induction we run agglomerative clustering and offer the user an interactive graphic visualization of the resulting dendogram in Figure 3. The user can explore different clusters settings and store those that represent the slots that he is interested in.

Once the event schemas have been created, we can use them to annotate documents. Figure 4 shows an example of an annotated document.

[1] For experiments we used the pre-trained 300 dimensional GoogleNews model from word2vec.

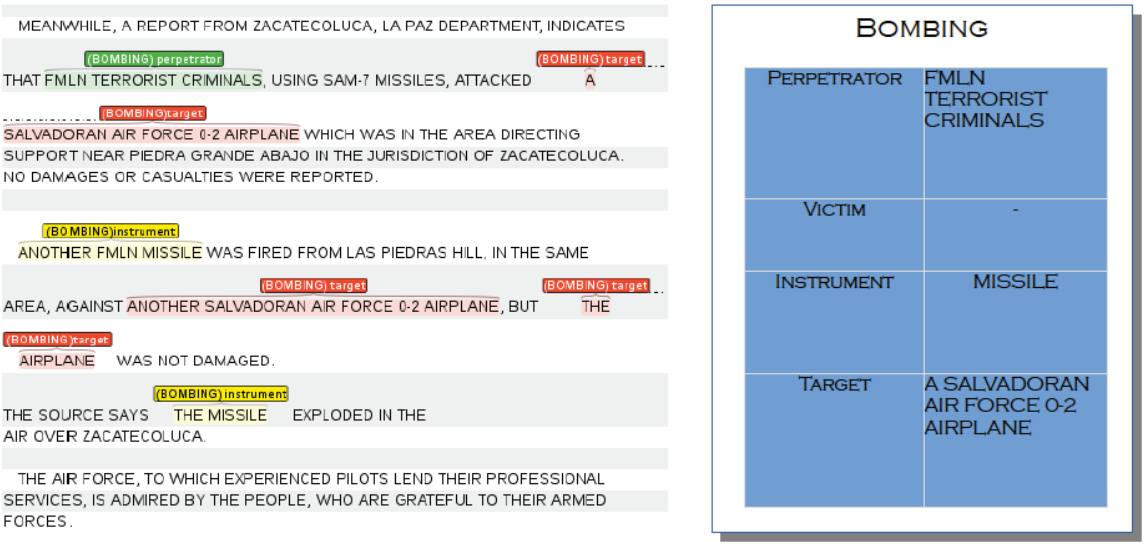


Figure 4: Slot Filling

4 Previous Work

To the best of our knowledge there is no previous work on interactive workflows for event schema induction. The most closely related work is on interactive relation extraction. Thilo and Alan (2015) presented a web toolkit for exploratory relation extraction that allows users to explore a corpus and build extraction patterns. Ralph and Yifan (2014) presented a system where users can create extractors for predifined entities and relations. Their approach is based on asking the user for seeding example instances which are then exploited with a semi-supervised learning algorithm. Marjorie et al. (2011) presented a system for interactive relation extraction based on active learning and boostrapping.

5 Conclusion

We have presented an interactive system for event schema induction, like in (Chambers and Jurafsky, 2011) the workflow is based on reducing the problem to two main clustering steps. Our system lets the user interact with the clustering process in a simple and intuitive manner and explore the corpus to create the schemas that better fits his information needs.

References

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. *arXiv preprint arXiv:1302.4813*.

Freedman Marjorie, Ramshaw Lance, Boschee Elizabeth, Gabbard Ryan, Kratkiewicz Gary, Ward Nicolas, and Weishedel Ralph. 2011. Extreme extraction: machine reading in a week. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics (ACL-15)*.

Grishman Ralph and He Yifan. 2014. An information extraction customizer. In *Proceedings of Text, Speech and Dialogue*.

Michael Thilo and Akbik Alan. 2015. Schnapper: A web toolkit for exploratory relation extraction. In *Proceedings of ACL-IJCNLP 2015 System Demostration (ACL-15)*.

ICE: Idiom and Collocation Extractor for Research and Education

Vasanthi Vuppuluri
Verizon Labs
375 W Trimble Rd,
San Jose, CA 95131, USA
vuppulurivasanthi@gmail.com

Shahryar Baki, An Nguyen, Rakesh Verma
Computer Science Dept.
University of Houston
Houston, TX 77204, USA
shahryar@cs.uh.com
anqnguyen@outlook.com
rverma@uh.edu

Abstract

Collocation and idiom extraction are well-known challenges with many potential applications in Natural Language Processing (NLP). Our experimental, open-source software system, called ICE, is a python package for flexibly extracting collocations and idioms, currently in English. It also has a competitive POS tagger that can be used alone or as part of collocation/idiom extraction. ICE is available free of cost for research and educational uses in two user-friendly formats. This paper gives an overview of ICE and its performance, and briefly describes the research underlying the extraction algorithms.

1 Introduction

Idioms and collocations are special types of phrases in many languages. An *idiom* is a phrase whose meaning cannot be obtained compositionally, i.e., by combining the meanings of the words that compose it. Collocations are phrases in which there is a semantic association between the component words and some restrictions on which words can be replaced and which cannot. In short, *collocations* are arbitrarily restricted lexeme combinations such as *look into* and *fully aware*.

Many scientists from diverse fields have worked on the challenging tasks of automated collocation and idiom extraction, e.g., see (Garg and Goyal, 2014; Seretan, 2013; Verma and Vuppuluri, 2015; Verma et al., 2016) and the references contained therein, yet there is no multi-purpose, ready-to-use, and flexible system for extracting these phrases. Collocation and its special forms, such as idioms, can be useful in many important tasks, e.g., summarization (Barrera and Verma, 2012), question-answering (Barrera et al., 2011), language translation, topic segmentation, authorial style, and so on. As a result, a tool for these tasks would be very handy.

To tackle this void, we introduce a feature-rich system called ICE (short for Idiom and Collocation Extractor), which has two versions: one is flexible and pipelined seamlessly for research purposes as a component of a larger system such as a question answering system, and the second as a web-based tool for educational purposes. ICE has a modular architecture and also includes a POS tagger, which can be used alone or as part of collocation or idiom extraction. An experiment with the CoNLL dataset shows that ICE's POS tagger is competitive against the Stanford POS tagger. For ease of use in research, we provide ICE as a python package.

For collocation extraction, ICE uses the IR models and techniques introduced by (Verma et al., 2016). These methods include: dictionary search, online dictionaries, a substitution method that compares the Bing hit counts of a phrase against the Bing hit counts of new phrases obtained by substituting the component words of the phrase one at a time to determine the "adherence factor" of the component words in a collocation, and two methods that try to measure the probability of association of the component words again using hit counts. In (Verma et al., 2016), the authors created a gold-standard dataset of collocations by taking 100 sentences at random from the Wiki50 dataset and manually annotating them for collocations (including idioms) using eight volunteers, who used the Oxford Dictionary of Collocations and Oxford Dictionary of Idioms. Each sentence was given to two annotators, who were given 25 sentences each for annotation, and their work was checked and corrected afterwards by two other people. In creating this dataset, even with the assistance of dictionaries, human performance

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 108–111

varied from an F1-score of about 39% to 70% for the collocation task. A comparison showed that their combination schemes outperformed existing techniques, such as MWEToolkit (Ramisch et al., 2010) and Text-NSP (Banerjee and Pedersen, 2003), with the best method achieving an F1-score of around 40% on the gold-standard dataset, which is within the range of human performance.

For idiom extraction, ICE uses the semantics-based methods introduced by (Verma and Vuppuluri, 2015). The salient feature of these methods is that they use Bing search for the definition of a given phrase and then check the compositionality of the phrase definition against combinations of the words obtained when a define word query is issued, where the word belongs to the phrase. If there is a difference in the meaning, that phrase is considered an idiom. In (Verma and Vuppuluri, 2015), authors showed that their method outperforms AMALGr (Schneider et al., 2014). Their best method achieved an F1-score of about 95% on the VNC tokens dataset.

Thus, ICE includes extraction methods for idioms and collocations that are state-of-the-art. Other tools exist for collocation extraction, e.g., see (Anagnostou and Weir, 2006), in which four methods including Text-NSP are compared.

2 ICE - Architecture and Algorithms

As ICE's algorithms are based on Bing search, users must provide a valid user id for the Bing API. ICE receives a list of sentences as an input and outputs a list of all collocations and idioms. It first splits the input sentences using NLTK sentence tokenizer, then generates n-grams and part of speech tags. ICE's n-gram generator takes care of punctuation marks and has been shown to be better than NSP's n-gram generator. Finally, the output n-grams are given to the collocation and idiom detection algorithms. Collocation and idiom extraction has been done by the algorithm given by (Verma et al., 2016)[1] and (Verma and Vuppuluri, 2015). For part of speech tagging we combined NLTK's regex tagger with NLTK's N-Gram Tagger to have a better performance on POS tagging. We compared our tagger with Stanford POS tagger (Manning et al., 2014) on the CoNLL dataset.[2] The accuracy of our tagger is 92.11%, which is slightly higher than 91.19%, the accuracy of the Stanford tagger on the same corpus.

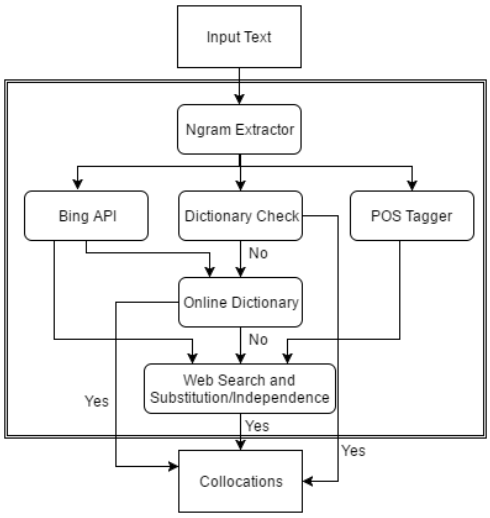


Figure 1: Collocation extractor diagram

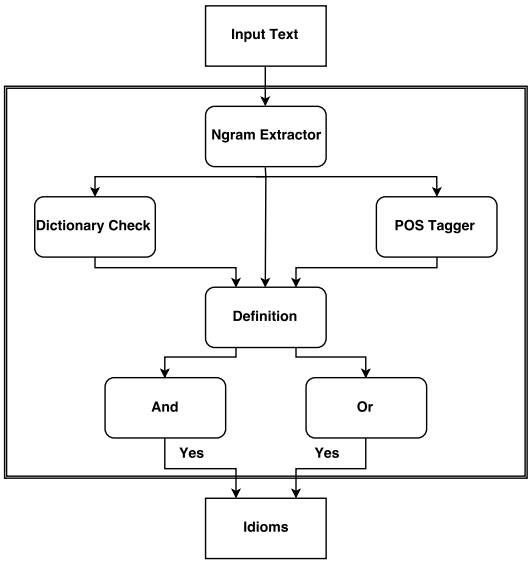


Figure 2: Idiom extractor diagram

Collocation/Idiom Extractor. The collocation extraction technique combines different methods in a pipeline in order to increase precision. Figures 1 and 2 show the idiom and collocation extraction system architectures separately. As shown in the diagrams, there are two methods for identifying idioms (called And and Or) and four different methods for identifying collocations including: offline dictionary search, online dictionary search, web search and substitution, and web search and independence.

For collocations, ICE pipelines the first and second methods, then pipelines them with the third or the fourth method (both options are available

[1] http://www2.cs.uh.edu/~rmverma/paper_216.pdf

[2] Available at http://www.cnts.ua.ac.be/conll2003/ner/

in the code). These methods are connected sequentially. This means that if something is considered as a collocation in one component, it will be added to the list of collocations and will not be given to the next component (yes/no arrows in the diagram). Table 1 shows the description of each component in collocation extractor diagram.

The *Ngram Extractor* receives all sentences and generates n-grams ranging from bigrams up to 8-grams. It uses NLTK sentence and word tokenizers for generating tokens. Then, it combines the generated tokens together taking care of punctuation to generate the n-grams.

Dictionary Check uses WordNet (Miller, 1995) as a dictionary and looks up each n-gram to see if it exists in WordNet or not (a collocation should exist in the dictionary). All n-grams that are considered as non-collocations are given to the next component as input.

The next component is *Online Dictionary*. It searches online dictionaries to see if the n-gram exists in any of them or not. It uses Bing Search API [3] to search for n-grams in the web.

Web Search and Substitution is the next component in the pipeline. This method uses Bing Search API to obtain hit counts for a phrase query. Then each word in the n-gram will be replaced by 5 random words (one at the time), and the hit counts are obtained. At the end, we will have a list of hit counts. These values will be used to differentiate between collocations and non-collocations.

The last component in the pipeline of collocation extraction is *Web Search and Independence*. The idea of this method is to check whether the probability of a phrase exceeds the probability that we would expect if the words are independent. It uses hit counts in order to estimate the probabilities. These probabilities are used to differentiate between collocations and non-collocations.

When running the collocation extraction function, one of the components should be selected out of the third and fourth ones.

The *Idiom Extractor* diagram is relatively simpler. Given the input n-gram, it creates n + 1 sets. The first contains (stemmed) words in the meaning of the phrase. The next n sets contain stemmed word in the meaning of each word in the n-gram. Then it applies the set difference operator to n pairs containing the first set and each of the n sets. The Or subsystem considers a phrase as an idiom if at least one word survives one of the subtractions (union of difference sets should be non-empty). For the And, at least one word has to exist that survived every subtraction (intersection of difference sets should be non-empty)

Performance. ICE outperforms both Text-NSP and MWEToolkit. On the gold-standard dataset, ICE's F1-score was 40.40%, MWE-Toolkit's F1-score was 18.31%, and Text-NSP had 18%. We also compared our idiom extraction with AMALGr method (Schneider et al., 2014) on their dataset and the highest F1-score achieved by ICE was 95% compared to 67.42% for AMALGr. For detailed comparison of ICE's collocation and idiom extraction algorithm with existing tools, please refer to (Verma et al., 2016) and (Verma and Vuppuluri, 2015).

Sample Code. Below is the sample code for using ICE's collocation extraction as part of a bigger system. For idiom extraction you can use IdiomExtractor class instead of collocationExtractor.

```
>> input=["he and Chazz duel
 with all keys on the line."]
>>from ICE import
 CollocationExtractor
>>extractor =
 CollocationExtractor.
 with_collocation_pipeline(
 "T1" , bing_key = "Temp",
 pos_check = False)
>> print(extractor.
 get_collocations_of_length(
  input, length = 3))
>> ["on the line"]
```

Educational Uses. ICE also has a web-based interface for demonstration and educational purposes. A user can type in a sentence into an input field and get a list of the idioms or collocations in the sentence. A screen-shot of the web-based interface is shown in Figure 3.[4]

3 Conclusion

ICE is a tool for extracting idioms and collocations, but it also has functions for part of speech

[3] http://datamarket.azure.com/dataset/bing/search

[4] The web interface is accessible through https://shahryarbaki.ddns.net/collocation/

Table 1: Components of Collocation Extraction Subsystem of ICE

Component	Description
Ngram Extractor	Generates bigram up to 8gram
Dictionary Check	Look up ngram in dictionary (Wordnet)
Online Dictionary	Look up ngram in online dictionaries (Bing)
Web Search and Substitution	Hitcount for phrase query and 5 generated queries by randomly changing the words in the ngram
Web Search and Independence	Probability of a phrase exceeds the probability that we would expect if the words are independent

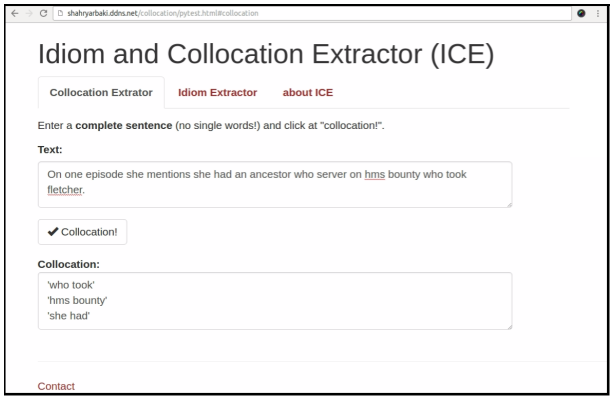


Figure 3: Screen-shot of the online version of ICE

tagging and n-gram extraction. All the components of the ICE are connected as a pipeline, hence every part of the system can be changed without affecting the other parts. The tool is available online at `https://github.com/shahryarabaki/ICE` as a python package and also at a website. The software is Licensed under the Apache License, Version 2.0.

References

Nikolaos K. Anagnostou and George R. S. Weir. 2006. Review of software applications for deriving collocations. In *ICT in the Analysis, Teaching and Learning of Languages, Preprints of the ICTATLL Workshop 2006*, pages 91–100.

Satanjeev Banerjee and Ted Pedersen. 2003. The design, implementation, and use of the ngram statistics package. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 370–381. Springer.

Araly Barrera and Rakesh Verma. 2012. Combining syntax and semantics for automatic extractive single-document summarization. In *CICLING*, volume LNCS 7182, pages 366–377.

Araly Barrera, Rakesh Verma, and Ryan Vincent. 2011. Semquest: University of houston's semantics-based question answering system. In *Proceedings of the Fourth Text Analysis Conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011*.

Chitra Garg and Lalit Goyal. 2014. Automatic extraction of idiom, proverb and its variations from text using statistical approach. *An International Journal of Engineering Sciences*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. Multiword expressions in the wild?: the mwetoolkit comes in handy. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 57–60. Association for Computational Linguistics.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Violeta Seretan. 2013. A multilingual integrated framework for processing lexical collocations. In *Computational Linguistics - Applications*, pages 87–108. Springer - Netherlands.

Rakesh Verma and Vasanthi Vuppuluri. 2015. A new approach for idiom identification using meanings and the web. *Recent Advances in Natural Language Processing*, pages 681–687.

Rakesh Verma, Vasanthi Vuppuluri, An Nguyen, Arjun Mukherjee, Ghita Mammar, Shahryar Baki, and Reed Armstrong. 2016. Mining the web for collocations: Ir models of term associations. In *International Conference on Intelligent Text Processing and Computational Linguistics*.

Bib2vec: Embedding-based Search System for Bibliographic Information

Takuma Yoneda **Koki Mori** **Makoto Miwa** **Yutaka Sasaki**
Department of Advanced Science and Technology
Toyota Technological Institute
2-12-1 Hisakata, Tempaku-ku, Nagoya, Japan
{sd14084,sd15435,makoto-miwa,yutaka.sasaki}@toyota-ti.ac.jp

Abstract

We propose a novel embedding model that represents relationships among several elements in bibliographic information with high representation ability and flexibility. Based on this model, we present a novel search system that shows the relationships among the elements in the ACL Anthology Reference Corpus. The evaluation results show that our model can achieve a high prediction ability and produce reasonable search results.

1 Introduction

Modeling relationships among several types of information, such as nodes in information network, has attracted great interests in natural language processing (NLP) and data mining (DM), since their modeling can uncover hidden information in data. Topic models such as author-topic model (Rosen-Zvi et al., 2004) have been widely studied to represent relationships among these types of information. These models, however, need a considerable effort to incorporate new types and do not scale well in increasing the number of types since they explicitly model the relationships between types in the generating process.

Word representation models, such as skip-gram and continuous bag-of-word (CBOW) (Mikolov et al., 2013), have made a great success in NLP. They have been widely used to represent texts, but recent studies started to apply these methods to represent other types of information, e.g., authors or papers in citation networks (Tang et al., 2015).

We propose a novel embedding model that represents relationships among several elements in bibliographic information, which is useful to discover hidden relationships such as authors' interests and similar authors. We built a novel search system that enables to search for authors and words related to other authors based on the model using the ACL Anthology Reference Corpus (Bird et al., 2008). Based on skip-gram and CBOW, our model embeds vectors to not only words but also other elements of bibliographic information such as authors and references and provides a great representation ability and flexibility. The vectors can be used to calculate distances among the elements using similarity measures such as cosine distance and inner products. For example, the distances can be used to find words or authors related to a specific author. Our model can easily incorporate new types without changing the model structure and scale well in the number of types.

2 Related works

Most previous work on modeling several elements in bibliographic information is based on topic models such as author-topic model (Rosen-Zvi et al., 2004). Although the models work fairly well, they have comparably low flexibility and scalability since they explicitly model the generation process. Our model employs word representation-based models instead of topic models.

Some previous work embedded vectors to the elements. Among them, large-scale information network embedding (LINE) (Tang et al., 2015) embedded a vector to each node in information network. LINE handles single type of information and prepares a network for each element separately. By contrast, our model simultaneously handles all the types of information.

3 Method

We propose a novel method to represent bibliographic information by embedding vectors to elements based on skip-gram and CBOW.

3.1 Task definition

We assume the bibliographic data set has the following structure. The data set is composed of bib-

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 112–115

liographic information of papers. Each paper consists of several categories. Categories are divided into two groups: a textual category Ψ (e.g., titles and abstracts[1]) and non-textual categories Φ (e.g., authors and references). Figure 1 illustrates an example structure of bibliographic information of a paper. Each category has one or more elements; the textual category usually has many elements while a non-textual category has a few elements (e.g., authors are not many for a paper).

3.2 Proposed model

Our model focuses on a *target* element, and predicts a *context* element from the target element. We use only the elements in non-textual categories as contexts to reduce the computational cost. Figure 1 shows the case when we use an element in a non-textual category as a target. For the black-painted target element in category Φ^2, the shaded elements in the same paper are used as its contexts.

When we use elements in the textual category as a target, instead of treating each element as a target, we consider that the textual category has only one element that represents all the elements in the category like CBOW. Figure 1 exemplifies the case that we consider the averaged vector of the vectors of all the elements in the textual category as a target.

We describe our probabilistic model to predict a context element e_O^j from a target e_I^i in a certain paper. We define two d-dimensional vectors v_t^i and ω_t^i to represent an element e_t^i as a target and context, respectively. Similarly to the skip-gram model, the probability to predict element e_O^j in the context from input e_I^i is defined as follows:

$$p(e_O^j|e_I^i) = \frac{\exp(\omega_O^j \cdot v_I^i + \beta_O^j)}{\sum_{(\omega_s^j,\beta_s^j)\in S^j} \exp(\omega_s^j \cdot v_I^i + \beta_s^j)}, \quad e_O^j \in \Phi,\ e_I^i \in \Psi \cup \Phi, \tag{1}$$

where β_s^j denotes a bias corresponds to ω_s^j, and S^j denotes pairs of ω_s^j and β_s^j that belong to a category Φ^j. As we mentioned, our model considers that the textual category Ψ has only one averaged vector. The vector v_{rep}^j can be described as:

$$v_{rep}^j = \frac{1}{n}\sum_{q=1}^{n} v_q^j,\ e^j \in \Psi \tag{2}$$

[1]Note that we have only one textual category since the categories for texts are usually not distinguished in most word representation models.

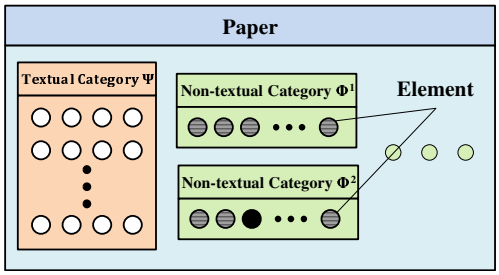


Figure 1: Example of the bibliographic information of a paper when the target is the element in the non-textual category. The black element is a target and the shaded elements are contexts.

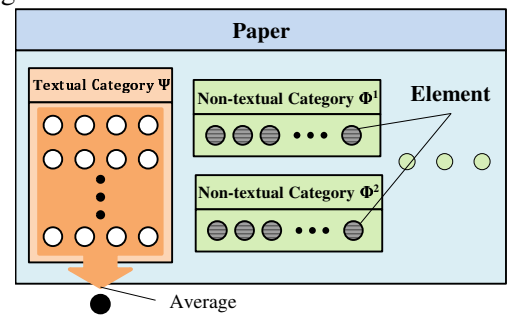


Figure 2: Example when the target is the elements in the textual category

Our target loss can be defined as:

$$-\sum_{(e_a,e_b)\in D} \log p(e_b|e_a), \tag{3}$$

where D denotes a set of all the correct pairs of the elements in the data set. To reduce the cost of the summation in Eq. (1), we applied the noise-contrastive estimation (NCE) to minimize the loss (Gutmann and Hyvärinen, 2010).

3.3 Predicting related elements

We predict the top k elements related to a query element by calculating their similarities to the query element. We calculate the similarities using one of three similarity measures: the linear function in Eq. (1), dot product, and cosine distance.

4 Experiments

4.1 Evaluation settings

We built our data set from the ACL Anthology Reference Corpus version 20160301 (Bird et al., 2008). The statistics of the data set and our model settings are summarized in Table 1.

As pre-processing, we deleted commas and periods that sticked to the tails of words and removed non-alphabetical words such as numbers

Category	Type	#Elements Original	Processed	Min. Freq.
text	textual	59,276	10,994	20
author	non-textual	17,260	2,609	5
reference	non-textual	10,871	10,871	1
year	non-textual	16	16	1
paper-id	non-textual	19,475	19,475	1

Table 1: Summary of our data set and model

and brackets from abstracts and titles. We then lowercased the words, and made phrases using the word2phrase tool[2].

We prepared 5 categories: *author*, *paper-id*, *reference*, *year* and *text*. *author* consists of the list of authors without distinguishing the order of the authors. *paper-id* is an unique identifier assigned to each paper, and this mimics the paragraph vector model (Le and Mikolov, 2014). *reference* includes the paper ids of reference papers in this data set. Although ids in paper-id and reference are shared, we did not assign the same vectors to the ids since they are different categories. *year* is the publication year of the paper. *text* includes words and phrases in both abstracts and titles, and it belongs to the textual category Ψ, while each other category is treated as a non-textual category Φ^i. We regard elements as unknown elements when they appear less than minimum frequencies in Table 1.

We split the data set into training and test. We prepared 17,475 papers for training and the remaining 2,000 papers for evaluation. For the test set, we regarded the elements that do not appear in the training set as unknown elements.

We set the dimension d of vectors to 300 and show the results with the linear function.

4.2 Evaluation

We automatically built multiple choice questions and evaluate the accuracy of our model. We also compared some results of our model with those of author-topic model.

Our method models elements in several categories and allows us to estimate relationships among the elements with high flexibility, but this makes the evaluation complex. Since it is tough to evaluate all the possible combinations of inputs and targets, we focused on relationships between authors and other categories. We prepared an evaluation data set that requires to estimate an author from other elements. We removed an (not unknown) author from each paper in the evaluation set to ask the system to predict the removed author considering all the other elements in the paper. To choose a correct author from all the authors can be insanely difficult, so we prepared 10 selection candidates. In order to evaluate the effectiveness of our model, we compared the accuracy on this data set with that by logistic regression. As a result, when we use our model, we got 74.3% (1,486 / 2,000) in accuracy, which was comparable to 74.1% (1,482 / 2,000) by logistic regression.

Table 2 shows the examples of the search results using our model. The leftmost column shows the authors we input to our model. In the rightmost two columns, we manually picked up words and authors belonging to a certain topic described in Sim et al. (2015) that can be considered to correspond to the input author. This table shows that our model can predict relative words or similar authors favorably well although the words are inconsistent with those by the author topic model.

Figure 3 shows the screenshot of our system. The lefthand box shows words in the word cloud related to the query and the righthand box shows the close authors. We can input a query by putting it in the textbox or click one of the authors in the righthand box and select a similarity measure by selecting a radio button.

4.3 Discussion

When we train the model, we did not use elements in category Ψ as context. This reduced the computational costs, but this might disturbed the accuracy of the embeddings. Furthermore, we used the averaged vector for the textual category Ψ, so we do not consider the importance of each word. Our model might ignore the inter-dependency among elements since we applied skip-grams. To resolve these problems, we plan to incorporate attentions (Ling et al., 2015) so that the model can pay more attentions to certain elements that are important to predict other elements.

We also found that some elements have several aspects. For example, words related to an author spread over several different tasks in NLP. We may be able to model this by embedding multiple vectors (Neelakantan et al., 2014).

5 Conclusions

This paper proposed a novel embedding method that represents several elements in bibliographic information with high representation ability and

[2] https://github.com/tmikolov/word2vec

	Our Model		Author Topic-Model	
Input Author	Relevant Words	Similar Authors	Topic Words	Topic Authors
Philipp Koehn	machine translation	Hieu Hoang	alignment	Chris Dyer
	hmeant	Alexandra Birch	translation	Qun Liu
	human translators	Eva Hasler	align	Hermann Ney
Ryan McDonald	dependency parsing	Keith Hall	parse	Michael Collins
	extrinsic	Slav Petrov	sentense	Joakim Nivre
	hearing	David Talbot	parser	Jens Nilson

Table 2: Working examples of our model and author topic-model

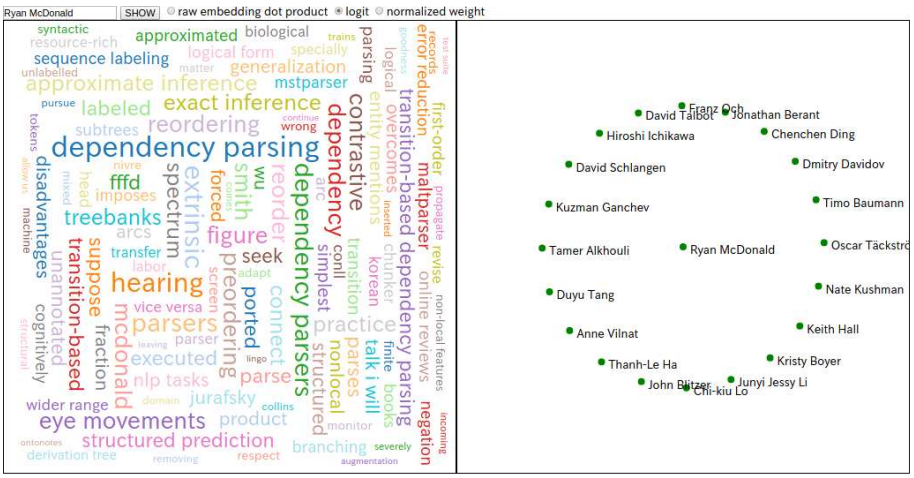


Figure 3: Screen shot of the system with the search results for the query "Ryan McDonald".

flexibility, and presented a system that can search for relationships among the elements in the bibliographic information. Experimental results in Table 2 show that our model can predict relative words or similar authors favorably well. We plan to extend our model by other modifications such as incorporating attention and embedding multiple vectors to an element. Since this model has high flexibility and scalability, it can be applied to not only papers but also a variety of bibliographic information in broad fields.

Acknowledgments

We would like to thank the anonymous reviewer for helpful comments and suggestions.

References

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan R. Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *LREC*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196.

Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso, and Chu-Cheng Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *EMNLP*, pages 1367–1372.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, pages 1059–1069.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *UAI*, pages 487–494.

Yanchuan Sim, Bryan R. Routledge, and Noah A. Smith. 2015. A utility model of authors in the scientific community. In *EMNLP*, pages 1510–1519.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: large-scale information network embedding. In *WWW*, pages 1067–1077.

The SUMMA Platform Prototype

Renars Liepins[†]
renars.liepins@leta.lv

Ulrich Germann[♂]
ugermann@inf.ed.ac.uk

Guntis Barzdins[†] · **Alexandra Birch**[♂] · **Steve Renals**[♂] · **Susanne Weber**[♀]
Peggy van der Kreeft[♁] · **Hervé Bourlard**[♁] · **João Prieto**[♁] · **Ondřej Klejch**[♂]
Peter Bell[♂] · **Alexandros Lazaridis**[♁] · **Alfonso Mendes**[♁] · **Sebastian Riedel**[♀]
Mariana S. C. Almeida[♁] · **Pedro Balage**[♁] · **Shay Cohen**[♂] · **Tomasz Dwojak**[♂]
Phil Garner[♁] · **Andreas Giefer**[♁] · **Marcin Junczys-Dowmunt**[♂] · **Hina Imran**[♁]
David Nogueira[♁] · **Ahmed Ali**[♁] · **Sebastião Miranda**[♁] · **Andrei Popescu-Belis**[♁]
Lesly Miculicich Werlen[♁] · **Nikos Papasarantopoulos**[♂] · **Abiola Obamuyide**[‡]
Clive Jones[♀] · **Fahim Dalvi**[♁] · **Andreas Vlachos**[‡] · **Yang Wang**[♁] · **Sibo Tong**[♁]
Rico Sennrich[♂] · **Nikolaos Pappas**[♁] · **Shashi Narayan**[♂] · **Marco Damonte**[♂]
Nadir Durrani[♁] · **Sameer Khurana**[♁] · **Ahmed Abdelali**[♁] · **Hassan Sajjad**[♁]
Stephan Vogel[♁] · **David Sheppey**[♀] · **Chris Hernon**[♀] · **Jeff Mitchell**[♀]

[†]Latvian News Agency [♂]University of Edinburgh [♁]Deutsche Welle [♀]BBC
[♁]Idiap Research Institute [♁]Priberam Informatica S.A. [♀]University College London
[‡]University of Sheffield [♁]Qatar Computing Research Institute

Abstract

We present the first prototype of the *SUMMA* Platform: an integrated platform for multilingual media monitoring. The platform contains a rich suite of low-level and high-level natural language processing technologies: automatic speech recognition of broadcast media, machine translation, automated tagging and classification of named entities, semantic parsing to detect relationships between entities, and automatic construction / augmentation of factual knowledge bases. Implemented on the *Docker* platform, it can easily be deployed, customised, and scaled to large volumes of incoming media streams.

1 Introduction

SUMMA (Scalable Understanding of Multilingual Media)[1] is a three-year *Research and Innovation Action* (February 2016 through January 2019), supported by the European Union's *Horizon 2020* research and innovation programme. *SUMMA* is developing a highly scalable, integrated web-based platform to automatically monitor an arbitrarily large number of public broadcast and web-based news sources.

Two concrete use cases and an envisioned third use case drive the project.

1.1 Monitoring of External News Coverage

BBC Monitoring, a division of the *British Broadcasting Corporation* (BBC), monitors a broad variety of news sources from all over the world on behalf of the BBC and external customers. About 300[2] staff journalists and analysts track TV, radio, internet, and social media sources in order to detect trends and changing media behaviour, and to flag breaking news events. A single monitoring journalist typically monitors four TV channels and several online sources simultaneously. This is about the maximum that any person can cope with mentally and physically. Assuming 8-hour shifts, this limits the capacity of *BBC Monitoring* to monitoring about 400 TV channels at any given time on average. At the same time, *BBC Monitoring* has access to about 13,600 distinct sources, including some 1,500 TV and 1,350 radio broadcasters. Automating the monitoring process not only allows the BBC to cover a broader

[1] www.summa-project.eu

[2] To be reduced to 200 by the end of March, 2017.

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 116–119

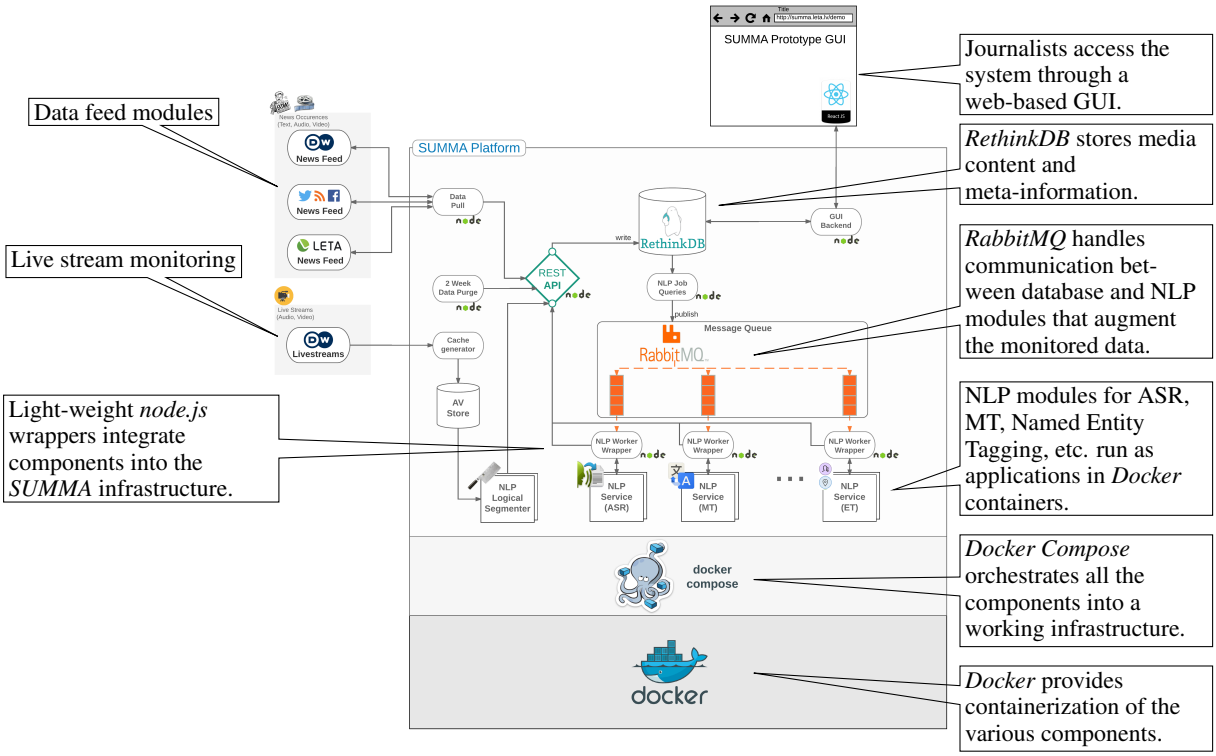


Figure 1: Architecture of the *SUMMA* Platform

spectrum of news sources, but also allows journalists to perform deeper analysis by enhancing their ability to search through broadcast media across languages in a way that other monitoring platforms do not support.

1.2 Monitoring Internal News Production

Deutsche Welle is Germany's international public service broadcaster. It provides international news and background information from a German perspective in 30 languages worldwide, 8 of which are used within *SUMMA*. News production within *Deutsche Welle* is organized by language and regional departments that operate and create content fairly independently. Interdepartmental collaboration and awareness, however, is important to ensure a broad, international perspective. Multilingual internal monitoring of world-wide news production (including underlying background research) helps to increase awareness of the work between the different language news rooms, decrease latency in reporting and reduce cost of news production within the service by allowing adaptation of existing news stories for particular target audiences rather than creating them from scratch.

1.3 Data Journalism

The third use case is data journalism. Measurable data is extracted from the content available in and produced by the *SUMMA* platform and graphics are created with such data. The data journalism dashboard will be able to provide, for instance, a graphical overview of trending topics over the past 24 hours or a heatmap of storylines. It can place geolocations of trending stories on a map. Customised dashboards can be used to follow particular storylines. For the internal monitoring use case, it will visualize statistics of content that was reused by other language departments.

2 System Architecture

Figure 1 shows an overview of the *SUMMA* Platform prototype. The Platform is implemented as an orchestra of independent components that run as individual containers on the *Docker* platform. This modular architecture gives the project partners a high level of independence in their development.

The system comprises the following individual processing components.

2.1 Data Feed Modules and Live Streams

These modules each monitor a specific news source for new content. Once new content is available, it is downloaded and fed into the database via a common REST API. Live streams are automatically segmented into logical segments.

2.2 Database Back-end

Rethink-DB[3] serves as the database back-end. Once new content is added, *Rethink-DB* issues processing requests to the individual NLP processing modules via *RabbitMQ*.

[3] `www.rethinkdb.com`

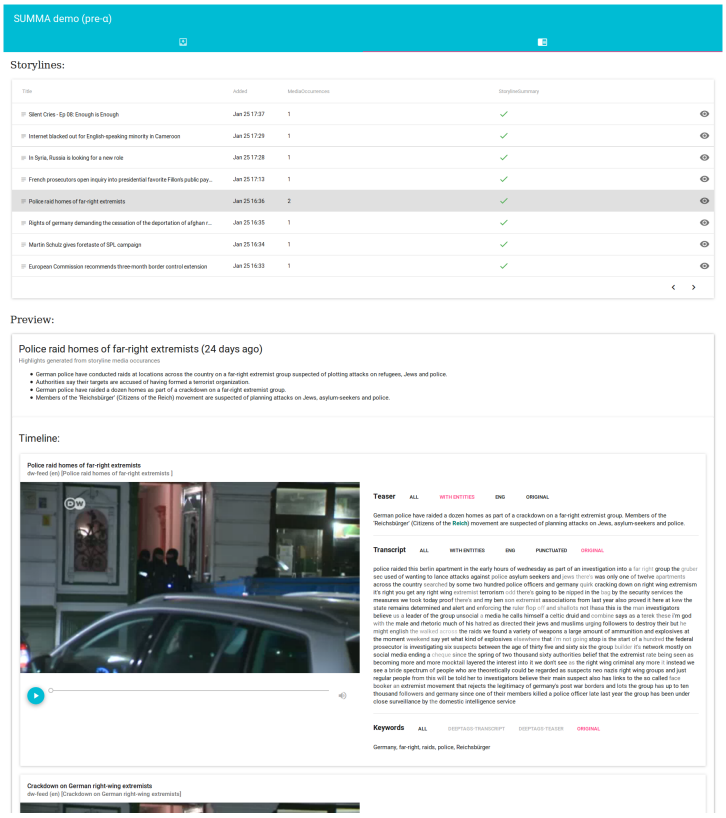


Figure 2: Web-based User Interface of the *SUMMA* Platform (Storyline View)

2.3 Automatic Speech Recognition

Spoken language from audio and video streams is first processed by automatic speech recognition to turn it into text for further processing. Models are trained on speech from the broadcast domain using the *Kaldi* toolkit (Povey et al., 2011); speech recognition is performed using the *CloudASR* platform (Klejch et al., 2015).

2.4 Machine Translation

The lingua franca within *SUMMA* is English. Machine translation based on neural networks is used to translate content into English automatically. The back-end MT systems are trained with the *Nematus* Toolkit (Sennrich et al., 2017); translation is performed with *AmuNMT* (Junczys-Dowmunt et al., 2016).

2.5 Entity Tagging and Linking

Depending on the source language, Entity Tagging and Linking is performed either natively, or on the English translation. Entities are detected with *TurboEntityRecognizer*, a named entity recognizer within *TurboParser*[4] (Martins et al., 2009). Then, we link the detected mentions to the knowledge base with a system based on our submission to TAC-KBP 2016 (Paikens et al., 2016).

[4] https://github.com/andre-martins/TurboParser

2.6 Topic Recognition and Labeling

This module labels incoming news documents and transcripts with a fine-grained set of topic labels. The labels are learned from a multilingual corpus of nearly 600k documents in 8 of the 9 *SUMMA* languages (all except Latvian), which were manually annotated by journalists at *Deutsche Welle*. The document model is a hierarchical attention network with attention at each level of the hierarchy, inspired by Yang et al. (2016), followed by a sigmoid classification layer.

2.7 Deep Semantic Tagging

The system also has a component that performs semantic parsing into Abstract Meaning Representations (Banarescu et al., 2013) with the aim to incorporate them into the storyline generation eventually. The parser was developed by Damonte et al. (2017). It is an incremental left-to-right parser that builds an AMR graph structure using a neural network controller. It also includes adaptations to German, Spanish, Italian and Chinese.

2.8 Knowledge Base Construction

This component provides a knowledge base of factual relations between entities, built with a model based on *Universal Schemas* (Riedel et al., 2013), a low-rank matrix factorization approach.The entity relations are extracted jointly across multiple languages, with entities pairs as rows and a set of structured relations and textual patterns as columns. The relations provide information about how various entities present in news

documents are connected.

2.9 Storyline Construction and Summarization

Storylines are constructed via online clustering, i.e., by assigning storyline identifiers to incoming documents in a streaming fashion, following the work in Aggarwal and Yu (2006). The resulting storylines are subsequently summarized via an extractive system based on Almeida and Martins (2013).

3 User Interface

Figure 2 shows the current web-based *SUMMA* Platform user interface in the *storyline* view. A storyline is a collection of news items that concerning a particular "story" and how it develops over time. Details of the layout are explained in the figure annotations.

4 Future Work

The current version of the Platform is a prototype designed to demonstrate the orchestration and interaction of the individual processing components. The look and feel of the page may change significantly over the course of the project, in response to the needs and requirements and the feedback from the use case partners, the BBC and *Deutsche Welle*.

5 Availability

The public release of the *SUMMA* Platform as open source software is planned for April 2017.

6 Acknowledgments

This work was conducted within the scope of the Research and Innovation Action *SUMMA*, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688139.

References

Charu C. Aggarwal and Philip S. Yu. 2006. A framework for clustering massive text and categorical data streams. In *SIAM Int'l. Conf. on Data Mining*, pages 479–483. SIAM.

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *ACL*, pages 196–206.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Linguistic Annotation Workshop*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *EACL*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. *CoRR*, abs/1610.01108.

Ondřej Klejch, Ondřej Plátek, Lukáš Žilka, and Filip Jurčíček. 2015. CloudASR: platform and service. In *Int'l. Conf. on Text, Speech, and Dialogue*, pages 334–341. Springer.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *ACL*, pages 342–350.

Peteris Paikens, Guntis Barzdins, Afonso Mendes, Daniel Ferreira, Samuel Broscheit, Mariana S. C. Almeida, Sebastião Miranda, David Nogueira, Pedro Balage, and André F. T. Martins. 2016. SUMMA at TAC knowledge base population task 2016. In *TAC*, Gaithersburg, Maryland, USA.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. 2011. The Kaldi speech recognition toolkit. In *ASRU*.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*, June.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *EACL Demonstration Session*, Valencia, Spain.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*, San Diego, CA, USA.

Author Index

Abdelali, Ahmed, 61, 116
Ai, Renlong, 5
Al Khatib, Khalid, 13
Ali, Ahmed, 61, 116
Almeida, Mariana S. C., 116

Baki, Shahryar, 108
Balage, Pedro, 116
Barzdins, Guntis, 116
Baumartz, Daniel, 17
Bell, Peter, 116
Bignotti, Enrico, 77
Birch, Alexandra, 65, 116
Boroş, Tiberiu, 33
Boullosa, Beto, 69
Bourlard, Herve, 116
Busemann, Stephan, 5

Cabrera, Benjamin, 21
Cartier, Emmanuel, 95
Cho, Kyunghyun, 65
Cohen, Shay B., 116
Cuadros, Montse, 49

da Cunha, Iria, 57
Dalvi, Fahim, 61, 116
Damonte, Marco, 116
Dehdari, Jon, 5
Dumitrescu, Stefan Daniel, 33
Durrani, Nadir, 61, 116
Dwojak, Tomasz, 116

Eckart de Castilho, Richard, 69

Ferrero, Germán, 104
Ferri Ramírez, Cesar, 41
Fersini, Elisabetta, 25
Firat, Orhan, 65

Gabryszak, Aleksandra, 5
Galitsky, Boris, 87
Gamallo, Pablo, 45
Garcia, Marcos, 45
Garner, Philip N., 116
Germann, Ulrich, 116
Geyken, Alexander, 69
Giefer, Andreas, 116
Gontrum, Johannes, 29
Groschwitz, Jonas, 29
Gurevych, Iryna, 69

Haddow, Barry, 65
Heigold, Georg, 5
Hemati, Wahed, 17
Hennig, Leonhard, 5
Henrich, Verena, 53
Hernon, Chris, 116
Hitschler, Julian, 65
Hysa, Luis, 57

Ilvovsky, Dmitry, 87
Imran, Hina, 116

Jones, Clive, 116
Junczys-Dowmunt, Marcin, 65, 116

Khurana, Sameer, 61, 116
Kiesel, Johannes, 13
Klejch, Ondrej, 116
Koller, Alexander, 29
Kutuzov, Andrey, 99
Kuzmenko, Elizaveta, 99

Lang, Alexander, 53
Läubli, Samuel, 65
Lazaridis, Alexandros, 116
Lemnitzer, Lothar, 69
Lepri, Bruno, 77
Liepins, Renars, 116
List, Johann-Mattis, 9

Manchanda, Pikakshi, 25
Marsi, Erwin, 91
Mehler, Alexander, 17
Mendes, Alfonso, 116
Menini, Stefano, 77
Messina, Enza, 25
Miceli Barone, Antonio Valerio, 65
Miculicich Werlen, Lesly, 116
Minock, Michael, 1

Miranda, Sebastião, 116
Mitchell, Jeff, 116
Miwa, Makoto, 112
Mokry, Jozef, 65
Montané, M. Amor, 57
Moreno-Ortiz, Antonio, 73
Moretti, Giovanni, 77
Mori, Koki, 112
Morillo Alcívar, Paulina Adriana, 41
Mubarak, Hamdy, 61

Nadejde, Maria, 65
Narayan, Shashi, 116
Nguyen, An, 108
Nogueira, David, 116
Nozza, Debora, 25

Obamuyide, Abiola, 116

Palmonari, Matteo, 25
Papasarantopoulos, Nikos, 116
Pappas, Nikolaos, 116
Perez, Naiara, 49
Pinar Øzturk, Pinar, 91
Pipa, Sonia, 33
Popescu-Belis, Andrei, 116
Prieto, João, 116
Primadhanty, Audi, 104

Quattoni, Ariadna, 104

Renals, Steve, 116
Rethmeier, Nils, 5
Riedel, Sebastian, 116
Ristagno, Fausto, 25
Rodríguez-Torres, Iván, 45
Ross, Björn, 21
Rubino, Raphael, 5

Sajjad, Hassan, 61, 116
Sasaki, Yutaka, 112
Schmeier, Sven, 5
Sennrich, Rico, 65, 116
Sheppey, David, 116
Sprugnoli, Rachele, 77
Steffen, Jörg, 5
Stein, Benno, 13
Steinert, Laura, 21

Teichmann, Christoph, 29
Thomas, Philippe, 5
Thorne, James, 37
Tonelli, Sara, 77
Tong, Sibo, 116
Torr, John, 81

Uslu, Tolga, 17
Uszkoreit, Hans, 5

V. Ardelan, Murat, 91
Vallejo Huanga, Diego Fernando, 41
van der Kreeft, Peggy, 116
van Genabith, Josef, 5
Verma, Rakesh, 108
Vlachos, Andreas, 37, 116
Vogel, Stephan, 61, 116
vuppuluri, vasanthi, 108

Wachsmuth, Henning, 13
Wang, He, 5
Wang, Yang, 116
Weber, Susanne, 116

Xu, Feiyu, 5

Yoneda, Takuma, 112

Zhang, Yifan, 61