EACL 2014

Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics

April 26-30, 2014
Gothenburg, Sweden

GOLD SPONSORS

Google™

معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute
Member of Qatar Foundation عضو في مؤسسة قطر

SILVER SPONSOR

FINDWISE
SEARCH DRIVEN SOLUTIONS

BRONZE SPONSORS

facebook

Yandex

SUPPORTERS

W
UNIVERSITY *of* WASHINGTON

UNIVERSITY OF GOTHENBURG
MASTER'S PROGRAMME IN
LANGUAGE TECHNOLOGY

Talkamatic
FREE DIALOGUE

EXHIBITORS

M&C
MORGAN & CLAYPOOL PUBLISHERS

Springer

OTHER SPONSORS

UNIVERSITY OF GOTHENBURG

City of Gothenburg

SAS OFFICIAL AIRLINE
A STAR ALLIANCE MEMBER

HOSTS

CLT
Centre for Language Technology
Gothenburg, Sweden

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 978-1-937284-75-6

Preface: General Chair

Dear readers,

Welcome to EACL 2014, the 14th Conference of the European Chapter of the Association for Computational Linguistics! This is the largest EACL meeting ever: with eighty long papers, almost fifty short ones, thirteen student research papers, twenty-six demos, fourteen workshops and six tutorials, we expect to bring to Gothenburg up to five hundred participants, for a week of excellent science interspersed with entertaining social events.

It is hard to imagine how much work is involved in the preparation of such an event. It takes about three years, from the day the EACL board starts discussing the location and nominating the chairs, until the final details of the budget are resolved. The number of people involved is also huge, and I was fortunate to work with an excellent, dedicated and efficient team, to which I am enormously grateful.

The scientific program was very ably composed by the Program Committee Chairs, Sharon Goldwater and Stefan Riezler, presiding over a team of twenty-four area chairs. Given that this year we had long paper submissions, followed by a rebuttal period, followed by a very stressed short paper reviewing period, this meant *a lot* of work. Overall, Sharon and Stefan handled over five hundred submissions, or over 1,500 reviews! The result of this work is a balanced, high-quality scientific program that I'm sure we will all enjoy. The PC Chairs have also selected the three invited speakers, and we will have the pleasure of attending keynotes delivered by Simon King, Ulrike von Luxburg, and Dan Roth – a great choice of speakers!

The diverse workshop program was put together by the Workshop Chairs, Anja Belz and Reut Tsarfaty, under very strict deadlines due to the fact that as in previous years, workshops were coordinated with other ACL events (this year, ACL and EMNLP). Even in light of the competition, Anja and Reut negotiated a varied and attractive set of fourteen workshops which will keep us busy over the weekend prior to the main conference.

Also on that weekend are the six tutorials, selected from among several submissions by the Tutorial Chairs, Afra Alishahi and Marco Baroni. Again, the tutorials offer a set of diverse and timely topics, covering both core areas of NLP and tangential fields of research.

We included in the program a large number of demonstrations, selected by Marko Tadić and Bogdan Babych, the Demo Chairs. And an integral part of the scientific program is the Student Research Workshop, put together by the SRW Chairs, Desmond Elliott, Konstantina Garoufi, Douwe Kiela, and Ivan Vulić, whose work was supervised by the SRW Faculty Advisor, Sebastian Padó.

The Proceedings that you're reading now were compiled by the Publication Chairs, Gosse Bouma and Yannick Parmentier. Their responsibilities include the preparation of all the proceedings, including the main session, the SRW, the demo session, the workshop proceedings etc. – thousands of pages, all under very strict deadlines.

It has been a very special pleasure for me to work with an excellent local organization team. The Local Organization Chairs, Lars Borin and Aarne Ranta, were assisted by an extremely efficient team, Yvonne Adesam, Martin Kaså and Nina Tahmasebi. Their effort cannot be overestimated: from dealing with the two universities over issues of conference space and funding, through dealing with two professional conference organizers, to corresponding with authors, participants and of course all the other chairs. Add the stress involved in being in charge of a hefty budget that has to be balanced by the end of the conference, and you can only admire the relaxed way in which they took upon themselves this daunting task.

The local team included also Peter Ljunglöf, the Publicity Chair, to whom we should all be grateful for the beautiful web site of the conference and the timely e-mails, tweets and Facebook statuses. The Local Sponsorship Chairs, Sofie Johansson Kokkinakis and Staffan Larsson, worked together with the ACL Sponsorship Chairs Jochen Leidner and Alessandro Moschitti, to obtain some much needed financial support. Sincere thanks are due to the various sponsors for their generous contribution.

The local team did a wonderful job organizing a social program this year. This includes a reception at the City Hall on Sunday, a catered poster and demo session on Monday, a conference dinner on Tuesday and of course, the famous Cortège at the very end of the conference. A perfect mix of business and pleasure.

I am grateful to all members of the EACL board for their advice and guidance, and in particular to past Chair Sien Moens, Chair Stephen Clark, Chair-elect Lluìs Màrquez and Treasurer Mike Rosner. Many thanks are also due to the ACL Treasurer Graeme Hirst and of course, as always, to the ACL Business Manager Priscilla Rasmussen, who was always there with her vast experience to clear up uncertainties and lend a helping hand.

Finally, let us not forget that this is all about *you*: authors, reviewers, demo presenters, workshop organizers and speakers, tutorial speakers and participants of the conference. Thank you for choosing to be part of EACL-2014, I wish you a very enjoyable conference!

Shuly Wintner, University of Haifa
General Chair
March 2014

Preface: Demo Chairs

In front of you there is a volume containing all contributions accepted for Demonstrations program at the EACL2014 conference which takes place from 26 to 30 April 2014 in Gothenburg, Sweden. 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. Probably for the first time in the line of many EACL conferences, the number of contributions for demonstations of new/existing software systems dealing with different types of language/speech processing, exceeded the number of available slots by almost three times. This confronted us with the situation where we were forced to introduce a strict selection process although the number of very good and excellent demo proposals was surprisingly high. The selection process was completed following the predefined criteria of relevance for EACL2014 conference, novelty, overall usability and applicability to more than one linguistic level and to more than one computational linguistic field/task. The contributions that had to be left below the threshold were not necessary of lower quality, but found themselves in more competitive environment this time. 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 demonstations 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
Marko Tadić
Bogdan Babych

Table of Contents

Demo Program

Session 1

Session 2

ITU Turkish NLP Web Service

Gülşen Eryiğit
Department of Computer Engineering
Istanbul Technical University
Istanbul, 34469, Turkey
gulsen.cebiroglu@itu.edu.tr

Abstract

We present a natural language processing (NLP) platform, namely the "ITU Turkish NLP Web Service" by the natural language processing group of Istanbul Technical University. The platform (available at tools.nlp.itu.edu.tr) operates as a SaaS (Software as a Service) and provides the researchers and the students the state of the art NLP tools in many layers: preprocessing, morphology, syntax and entity recognition. The users may communicate with the platform via three channels: 1. via a user friendly web interface, 2. by file uploads and 3. by using the provided Web APIs within their own codes for constructing higher level applications.

1 Introduction

ITU NLP research group is devoted to produce Turkish NLP tools for more than 10 years. The group offers many NLP courses in graduate level and core NLP research components to different research groups both in NLP field and other disciplines: e.g. linguistics, data mining, web mining and information retrieval. The motivation of the presented platform in this paper comes from the real word problems of sharing the produced NLP resources by different people from varying level of computer background (starting from undergraduates to PhD students or researchers, people from other fields (e.g.linguistics)). These may be categorized under the following main problems:

1. Need to provide assistance for the installation and the usage of different tools, all posing different technological requirements in the users' computers.
2. Difficulty to share the updates and the new modules introduced into the pipeline.
3. Difficulty of using the tools for educational purposes within the classrooms and term projects.
4. licensing issues of the underlying technologies (such as FST and machine learning softwares)

The difficulty in the ease-of-use of Turkish NLP tools and their inconsistencies with each others were also causing the replication of the same effort in different places and preventing the community from working on less-studied higher level areas for the Turkish language. A good example to this may be the efforts for creating Turkish morphological analyzers: some outstanding ones among many others are (Oflazer, 1994; Eryiğit and Adalı, 2004; Akın and Akın, 2007; Sak et al., 2008; Çöltekin, 2010; Şahin et al., 2013))

In this paper, we present our new web service which provides both a whole Turkish NLP pipeline (from raw data to syntax, example given in Figure1 priorly defined in (Eryiğit, 2012)) and its atomic NLP components for stand-alone usage, namely:

- Tokenizer
- Deasciifier
- Vowelizer
- Spelling Corrector
- Normalizer
- isTurkish
- Morphological Analyzer
- Morphological Disambiguator
- Named Entity Recognizer
- Dependency Parser

2 Provided Components

The provided components via our web service may be grouped under 4 layers: preprocessing, morphological processing, multiword expression handling and syntactic processing.

2.1 Preprocessing

The preprocessing layer consists of many sub components specifically developed for unformat-

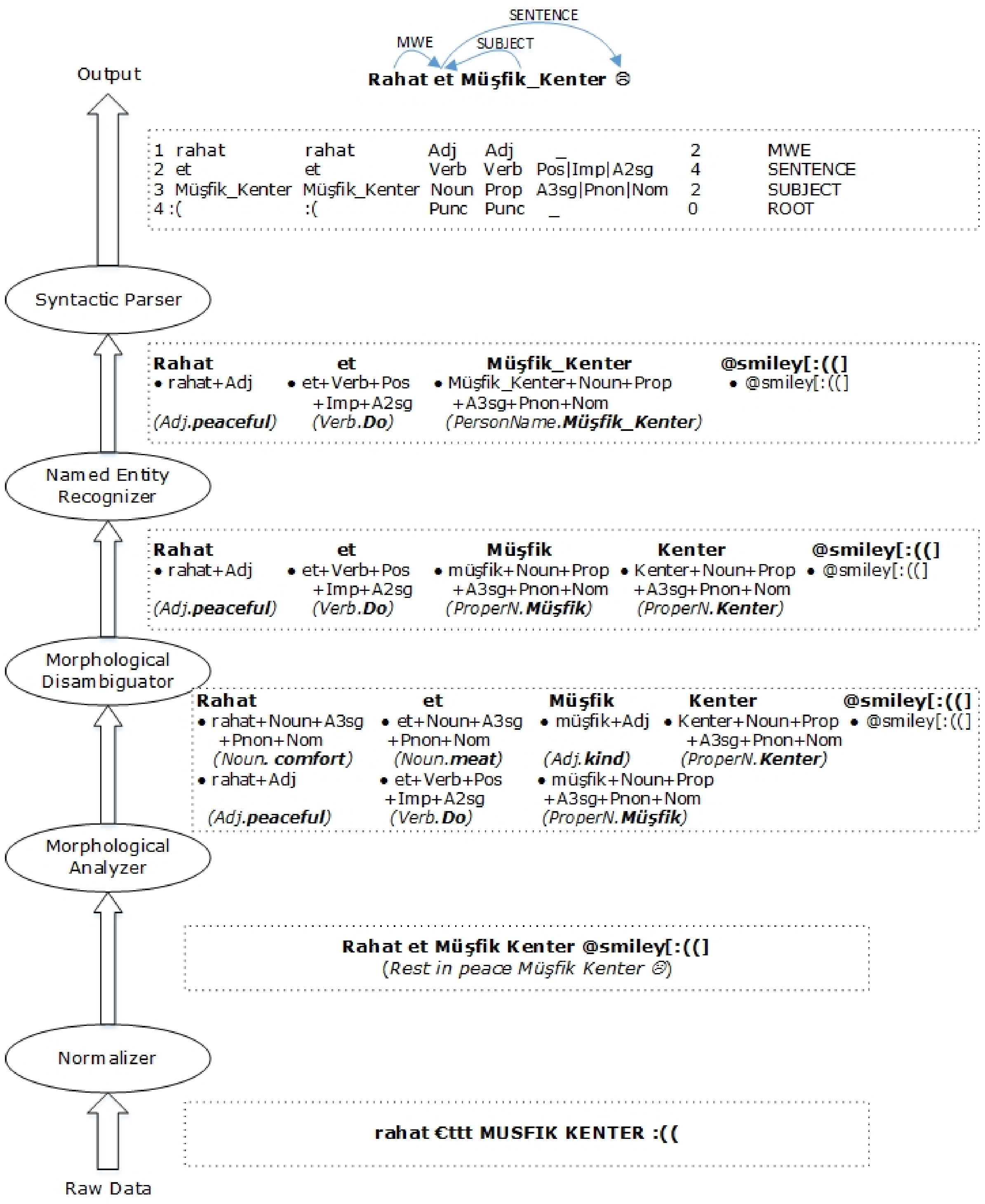

Output
SENTENCE
MWE
SUBJECT
Rahat et Müşfik_Kenter ☹
1 rahat rahat Adj Adj _ 2 MWE
2 et et Verb Verb Pos|Imp|A2sg 4 SENTENCE
3 Müşfik_Kenter Müşfik_Kenter Noun Prop A3sg|Pnon|Nom 2 SUBJECT
4 :(:(Punc Punc _ 0 ROOT
Syntactic Parser
Rahat • rahat+Adj (Adj.peaceful)
et • et+Verb+Pos +Imp+A2sg (Verb.Do)
Müşfik_Kenter • Müşfik_Kenter+Noun+Prop +A3sg+Pnon+Nom (PersonName.Müşfik_Kenter)
@smiley[:(() • @smiley[:(()
Named Entity Recognizer
Rahat • rahat+Adj (Adj.peaceful)
et • et+Verb+Pos +Imp+A2sg (Verb.Do)
Müşfik • müşfik+Noun+Prop +A3sg+Pnon+Nom (ProperN.Müşfik)
Kenter • Kenter+Noun+Prop +A3sg+Pnon+Nom (ProperN.Kenter)
@smiley[:(() • @smiley[:(()
Morphological Disambiguator
Rahat • rahat+Noun+A3sg +Pnon+Nom (Noun. comfort) • rahat+Adj (Adj.peaceful)
et • et+Noun+A3sg +Pnon+Nom (Noun.meat) • et+Verb+Pos +Imp+A2sg (Verb.Do)
Müşfik • müşfik+Adj (Adj.kind) • müşfik+Noun+Prop +A3sg+Pnon+Nom (ProperN.Müşfik)
Kenter • Kenter+Noun+Prop +A3sg+Pnon+Nom (ProperN.Kenter)
@smiley[:(() • @smiley[:(()
Morphological Analyzer
Rahat et Müşfik Kenter @smiley[:(()
(Rest in peace Müşfik Kenter ☹)
Normalizer
rahat €ttt MUSFIK KENTER :((
Raw Data

Figure 1: ITU Turkish NLP Pipeline

ted social media data in focus. These are a tokenizer, a diacritic restorer, a vowelizer, a spelling corrector and a normalizer. The diacritic restorer [1] is the component where the ASCII characters are transformed into their proper Turkish forms. The deasciifier (Adalı and Eryiğit, 2014) chooses the most probable candidate within the current context by using conditional random fields (CRFs). The **vocalizer** (Adalı and Eryiğit, 2014) restores the omitted vowels (generally within the social media messages for shortening purpose): e.g. "svyrm" will be converted to "seviyorum" (I love you). The **spelling corrector**[2] is kind of an adaptation of Wang et al.(2011) into agglutinative languages. The **normalizer** (Torunoğlu and Eryiğit, 2014) is constructed of the previous three components and many other modules and provides a state of the art text normalizer for Turkish.

2.2 Morphological Processing

This layer consists of a rule based **morphological analyzer** (Şahin et al., 2013; Şahin, 2014) which uses HFST-Helsinki Finite State Transducer (Lindén et al., 2009) and a hybrid **morphological disambiguator**[3]. This layer also provides the **isTurkish** component which validates a word by using the morphological analyzer.

2.3 Multi Word Expressions

As shown in Eryigit et al. (2011), the detection and unification of the named entities has the highest impact for the syntactic layer. That is why the following Turkish named entity recognizer (Şeker and Eryiğit, 2012) is included within the pipeline and the remaining multiword expressions are detected in the syntactic layer as shown in Figure 1 (dependency label MWE).

2.4 Syntactic Parsing

For the syntactic layer we are providing the state of the art dependency parser for Turkish presented in (Eryiğit et al., 2008; Nivre et al., 2007) which produces the ouputs in Conll format (Buchholz and Marsi, 2006).

3 Conclusion and Future Work

We introduced our ITU Turkish NLP Web Platform which provides us easier administration, automatic updates and patch management, compatibility, easier usage, easier collaboration[4] and global accessibility by being designed as a SaaS. Any body from any discipline with any level of underlying computer background may easily use our web interface either for only analyzing language data or for constructing more complicated NLP systems. The platform already attracted many users from different universities in Turkey and it is now started to get used in many research projects and graduate theses. We believe as being the pioneer serving almost all of the available and top performing NLP tools for Turkish, ITU Turkish NLP Web Service will fed light to new research topics for this language.

For now, the pipeline is constructed by converting the input output formats of each individual tools. But our current effort is to transform the platform into a UIMA(Ferrucci and Lally, 2004) compliant architecture so that it can also integrate with other such platforms universally. We also plan to service the new version of ITU Data Annotation Tool (Eryiğit, 2007) from the same address where the users will also be able to see their data visually (e.g. dependency trees)

Acknowledgments

I want to thank my students without whose it would be impossible to produce the ITU Turkish NLP pipeline: Thomas Joole, Dilara Torunoğlu, Umut Sulubacak and Hasan Kaya. This work is part of a research project supported by TUBITAK 1001(Grant number: 112E276) as an ICT cost action (IC1207) project.

References

Kübra Adalı and Gülşen Eryiğit. 2014. Vowel and diacritic restoration for social media texts. In *5th Workshop on Language Analysis for Social Media (LASM) at EACL*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for turkic languages. *Structure*.

Sabine Buchholz and Erwin Marsi. 2006. Conll-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164, New York, NY. Association for Computational Linguistics.

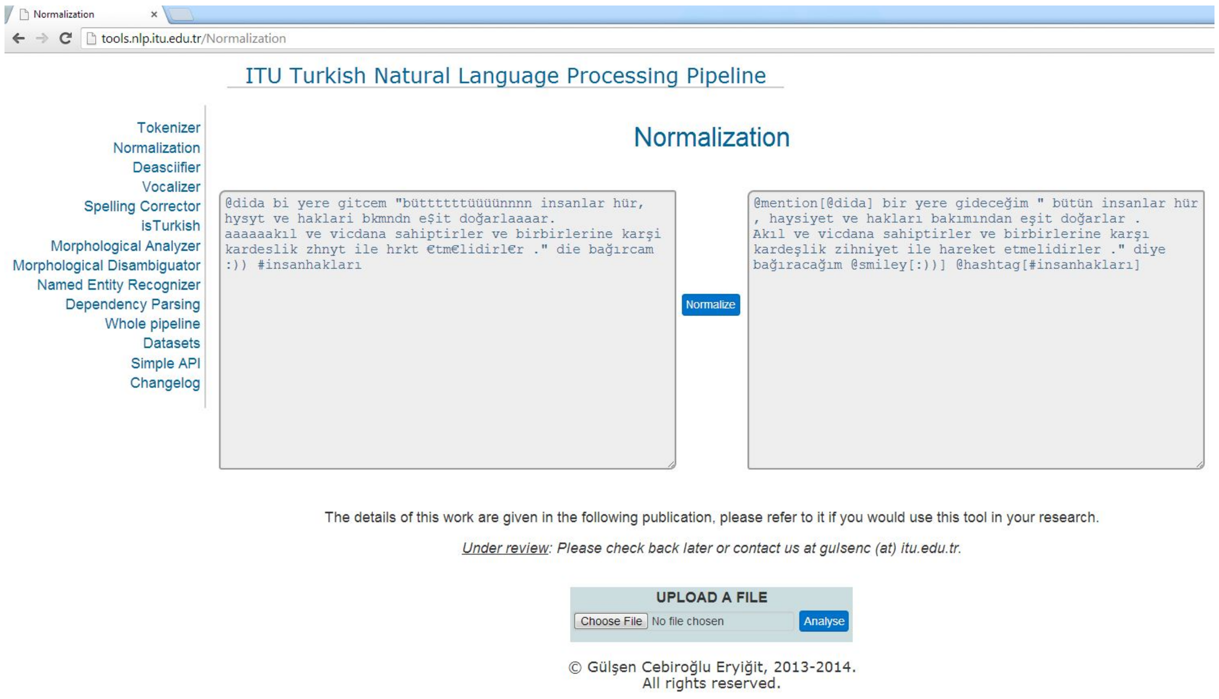
Çağrı Çöltekin. 2010. A freely available morphological analyzer for Turkish. In *Proceedings of the 7th International conference on Language Resources and Evaluation (LREC2010)*, pages 820–827.

[1]named as "**deasciifier**" since the term is already adopted by the Turkish community

[2]Publication in preparation.

[3]Publication in preparation.

[4]The mailing list notifications are sent to registered users with each new broadcast.



```
@dida bi yere gitcem "bütttttttüüüünnnn insanlar hür,
hysyt ve hakları bkmndn e$it doğarlaaaar.
aaaaaakıl ve vicdana sahiptirler ve birbirlerine karşi
kardeslik zhnyt ile hrkt €tm€lidirl€r ." die bağırcam
:)) #insanhakları
```

```
@mention[@dida] bir yere gideceğim " bütün insanlar hür
, haysiyet ve hakları bakımından eşit doğarlar .
Akıl ve vicdana sahiptirler ve birbirlerine karşı
kardeşlik zihniyet ile hareket etmelidirler ." diye
bağıracağım @smiley[:))] @hashtag[#insanhakları]
```

The details of this work are given in the following publication, please refer to it if you would use this tool in your research.

Under review: Please check back later or contact us at gulsenc (at) itu.edu.tr.

© Gülşen Cebiroğlu Eryiğit, 2013-2014.
All rights reserved.

Figure 2: ITU Turkish NLP Web Interface

Gülşen Eryiğit and Eşref Adalı. 2004. An affix stripping morphological analyzer for Turkish. In *Proceedings of the International Conference on Artificial Intelligence and Applications*, pages 299–304, Innsbruck, 16-18 February.

Gulsen Eryigit, Tugay Ilbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages (IWPT)*, pages 45–55, Dublin, Ireland, October. Association for Computational Linguistics.

Gülşen Eryiğit. 2007. Itu treebank annotation tool. In *Proceedings of the ACL workshop on Linguistic Annotation (LAW 2007)*, Prague, 24-30 June.

Gülşen Eryiğit. 2012. The impact of automatic morphological analysis & disambiguation on dependency parsing of turkish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey, 23-25 May.

Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of Turkish. *Computational Linguistics*, 34(3):357–389.

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.

Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. 2009. Hfst tools for morphology–an efficient open-source package for construction of morphological analyzers. In *State of the Art in Computational Morphology*, pages 28–47. Springer.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Stetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering Journal*, 13(2):99–135.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Muhammet Şahin, Umut Sulubacak, and Gülşen Eryiğit. 2013. Redefinition of Turkish morphology using flag diacritics. In *Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013)*, Phuket, Thailand, October.

Muhammet Şahin. 2014. ITUMorph, a more accurate and faster wide coverage morphological analyzer for Turkish. Master's thesis, Istanbul Technical University.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *GoTAL 2008*, volume 5221 of *LNCS*, pages 417–427. Springer.

Gökhan Akın Şeker and Gülşen Eryiğit. 2012. Initial explorations on using CRFs for Turkish named entity recognition. In *Proceedings of COLING 2012*, Mumbai, India, 8-15 December.

Dilara Torunoğlu and Gülşen Eryiğit. 2014. A cascaded approach for social media text normalization of Turkish. In *5th Workshop on Language Analysis for Social Media (LASM) at EACL*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ziqi Wang, Gu Xu, Hang Li, and Ming Zhang. 2011. A fast and accurate method for approximate string search. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 52–61.

Multilingual, Efficient and Easy NLP Processing with IXA Pipeline

Rodrigo Agerri
IXA NLP Group
Univ. of the Basque Country
UPV/EHU
Donostia San-Sebastián
rodrigo.agerri@ehu.es

Josu Bermudez
Deusto Institute of Technology
Deustotech
Univ. of Deusto
Bilbao
josu.bermudez@deusto.es

German Rigau
IXA NLP Group
Univ. of the Basque Country
UPV/EHU
Donostia-San Sebastián
german.rigau@ehu.es

Abstract

IXA pipeline is a modular set of Natural Language Processing tools (or pipes) which provide easy access to NLP technology. It aims at lowering the barriers of using NLP technology both for research purposes and for small industrial developers and SMEs by offering robust and efficient linguistic annotation to both researchers and non-NLP experts. IXA pipeline can be used "as is" or exploit its modularity to pick and change different components. This paper describes the general data-centric architecture of IXA pipeline and presents competitive results in several NLP annotations for English and Spanish.

1 Introduction

Many Natural Language Processing (NLP) applications demand some basic linguistic processing (Tokenization, Part of Speech (POS) tagging, Named Entity Recognition and Classification (NER), Syntactic Parsing, Coreference Resolution, etc.) to be able to further undertake more complex tasks. Generally, NLP annotation is required to be as accurate and efficient as possible and existing tools, quite righly, have mostly focused on performance. However, this generally means that NLP suites and tools usually require researchers to do complex compilation/installation/configuration in order to use such tools. At the same time, in the industry, there are currently many Small and Medium Enterprises (SMEs) offering services that one way or another depend on NLP annotations.

In both cases, in research and industry, acquiring, deploying or developing such base qualifying technologies is an expensive undertaking that redirects their original central focus: In research, much time is spent in the preliminaries of a particular research experiment trying to obtain the required basic linguistic annotation, whereas in an industrial environment SMEs see their already limited resources taken away from offering products and services that the market demands. IXA pipeline provides *ready to use modules* to perform efficient and accurate linguistic annotation to allow users to focus on their original, central task. When designing the architecture, we took several decisions with respect to what IXA pipeline had to be:

Simple and ready to use: Every module of the IXA pipeline can be up an running after two simple steps.

Portable: The modules come with "all batteries included" which means that no classpath configurations or installing of any third-party dependencies is required. The modules will will run on any platform as long as a JVM 1.7+ and/or Python 2.7 are available.

Modular: Unlike other NLP toolkits, which often are built in a monolithic architecture, IXA pipeline is built in a data centric architecture so that modules can be picked and changed (even from other NLP toolkits). The modules behave like Unix pipes, they all take standard input, do some annotation, and produce standard output which in turn is the input for the next module. The data-centric architecture of IXA pipeline means that any module is *highly independent* and can therefore be used with other tools from other toolkits if required.

Efficient: Piping the tokenizer (250K words per second) POS tagger and lemmatizer all in one process annotates over 5K words/second. The NERC module annotates over 5K words/second. In a multi-core machine, these times are dramatically reduced due to multi-threading.

Multilingual: Currently we offer NLP annotations for both English and Spanish, but other languages are being included in the pipeline. Tokenization already works for several languages, including Dutch, French, Italian, German, Spanish and English.

Accurate: For example, POS tagging and NERC for English and Spanish are comparable with other state of the art systems, as it is the coreference resolution module for English.

Apache License 2.0: IXA Pipeline is licensed under the Apache License 2.0, an open-source license that facilitates source code use, distribution and integration, also for commercial purposes.[1]

Next section describes the *IXA pipeline* architecture, section 3 the modules so far developed. Whenever available, we also present empirical evaluation. Section 4 describes the various ways of using the tools. Finally, section 5 discusses some concluding remarks.

[1]http://www.apache.org/licenses/LICENSE-2.0.html

2 Architecture

IXA pipeline is primarily conceived as a set of *ready to use tools* that can provide efficient and accurate linguistic annotation without any installation/configuration/compilation effort. As in Unix-like operative systems, IXA pipeline consists of a set of processes chained by their standard streams, in a way that the output of each process feeds directly as input to the next one. The Unix pipeline metaphor has been applied for NLP tools by adopting a very simple and well known data centric architecture, in which every module/pipe is interchangeable for another one as long as it takes and produces the required data format.

The data format in which both the input and output of the modules needs to be formatted to represent and filter linguistic annotations is KAF (Bosma et al., 2009). KAF is a language neutral annotation format representing both morpho-syntactic and semantic annotation in a structured format. KAF was originally designed in the Kyoto European project[2], but it has since been in continuous development[3]. Our Java modules all use kaflib[4] library for easy integration.

Every module in the IXA pipeline, except the coreference resolution, is implemented in Java, and requires Java JDK1.7+ to compile. The integration of the Java modules in the IXA pipeline is performed using Maven[5]. Maven is used to take care of classpaths configurations and third-party tool dependencies. This means that the binaries produced and distributed *will work off-the-self*. The coreference module uses pip[6] to provide an easy, one step installation. If the source code of an ixa-pipe-$module is cloned from the remote repository, *one command* to compile and have ready the tools will suffice.

Some modules in IXA pipeline provide linguistic annotation based on probabilistic supervised approaches such as POS tagging, NER and Syntactic Parsing. IXA pipeline uses two well known machine learning algorithms, namely, Maximum Entropy and the Perceptron. Both Perceptron (Collins, 2002; Collins, 2003) and Maximum Entropy models (Ratnaparkhi, 1999) are adaptable algorithms which have been successfully applied to NLP tasks such as POS tagging, NER and Parsing with state of the art results. To avoid duplication of efforts, IXA pipeline uses the already available open-source Apache OpenNLP API[7] to train POS, NER and parsing probabilistic models using these two approaches.

[2]http://kyoto-project.eu

[3]http://www.opener-project.org/kaf/

[4]https://github.com/ixa-ehu/kaflib

[5]http://maven.apache.org/

[6]https://pypi.python.org/pypi/pip

[7]http://opennlp.apache.org

3 Pipes

IXA pipeline currently provides the following linguistic annotations: Sentence segmentation, tokenization, Part of Speech (POS) tagging, Lemmatization, Named Entity Recognition and Classification (NER), Constituent Parsing and Coreference Resolution. Every module works for English and Spanish and is implemented in Java/Maven as described above. The only exception is the coreference resolution module, which currently is available in Python 2.7 and for English only (Spanish version will comme soon). We will now describe which annotation services are provided by each module of the pipeline.

3.1 ixa-pipe-tok

This module provides rule-based Sentence Segmentation and Tokenization for French, Dutch, English, Italian and Spanish. It produces tokenized and segmented text in KAF, running text and CoNLL formats. The rules are originally based on the Stanford English Tokenizer[8], but with substantial modifications and additions. These include tokenization for other languages such as French and Italian, normalization according the Spanish Ancora Corpus (Taulé et al., 2008), paragraph treatment, and more comprehensive gazeteers of non breaking prefixes. The tokenizer depends on a JFlex[9] specification file which compiles in seconds and performs at a very reasonable speed (around 250K word/second, and much quicker with Java multithreading).

3.2 ixa-pipe-pos

ixa-pipe-pos provides POS tagging and lemmatization for English and Spanish. We have obtained the best results so far with the same featureset as in Collins's (2002) paper. *Perceptron* models for English have been trained and evaluated on the WSJ treebank using the usual partitions (e.g., as explained in Toutanova et al. (2003). We currently obtain a performance of 97.07% vs 97.24% obtained by Toutanova et al., (2003)). For Spanish, *Maximum Entropy* models have been trained and evaluated using the Ancora corpus; it was randomly divided in 90% for training and 10% for testing. This corresponds to 440K words used for training and 70K words for testing. We obtain a performance of 98.88% (the corpus partitions are available for reproducibility). Giménez and Marquez (2004) report 98.86%, although they train and test on a different subset of the Ancora corpus.

Lemmatization is currently performed via 3 different dictionary lookup methods: (i) *Simple Lemmatizer*: It is based on HashMap lookups on a plain text dictionary. Currently we use dictionaries from the LanguageTool project[10] under their distribution licenses. The English

[8]http://www-nlp.stanford.edu/software/tokenizer.shtml

[9]http://jflex.de/

[10]http://languagetool.org/

dictionary contains 300K lemmas whereas the Spanish provides over 600K; (ii) *Morfologik-stemming*[11]: The Morfologik library provides routines to produce binary dictionaries, from dictionaries such as the one used by the Simple Lemmatizer above, as finite state automata. This method is convenient whenever lookups on very large dictionaries are required because it reduces the memory footprint to 10% of the memory required for the equivalent plain text dictionary; and (iii) We also provide lemmatization by lookup in *WordNet-3.0* (Fellbaum and Miller, 1998) via the JWNL API[12]. Note that this method is only available for English.

3.3 ixa-pipe-nerc

Most of the NER systems nowdays consist of language independent systems (sometimes enriched with gazeteers) based on automatic learning of statistical models. *ixa-pipe-nerc* provides Named Entity Recognition (NER) for English and Spanish. The named entity types are based on the CONLL 2002[13] and 2003[14] tasks which were focused on language-independent supervised named entity recognition (NER) for four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. We currently provide two very fast language independent models using a rather simple baseline featureset (e.g., similar to that of Curran and Clark (2003), except POS tag features).

For English, perceptron models have been trained using CoNLL 2003 dataset. We currenly obtain 84.80 F1 which is coherent with other results reported with these features (Clark and Curran, 2003; Ratinov and Roth, 2009). The best Stanford NER model reported on this dataset achieves 86.86 F1 (Finkel et al., 2005), whereas the best system on this dataset achieves 90.80 F1 (Ratinov and Roth, 2009), using non local features and substantial external knowledge.

For Spanish we currently obtain best results training Maximum Entropy models on the CoNLL 2002 dataset. Our best model obtains 79.92 F1 vs 81.39 F1 (Carreras et al., 2002), the best result so far on this dataset. Their result uses external knowledge and without it, their system obtains 79.28 F1.

3.4 ixa-pipe-parse

ixa-pipe-parse provides statistical constituent parsing for English and Spanish. Maximum Entropy models are trained to build shift reduce bottom up parsers (Ratnaparkhi, 1999) as provided by the Apache OpenNLP API. Parsing models for English have been trained using the Penn treebank and for Spanish using the Ancora corpus (Taulé et al., 2008).

Furthermore, *ixa-pipe-parse* provides two methods of HeadWord finders: one based on Collins' head rules as defined in his PhD thesis (1999), and another one based on Stanford's parser Semantic Head Rules[15]. The latter are a modification of Collins' head rules according to lexical and semantic criteria. These head rules are particularly useful for the Coreference resolution module and for projecting the constituents into dependency graphs.

As far as we know, and although previous approaches exist (Cowan and Collins, 2005), *ixa-pipe-parse* provides the first publicly available statistical parser for Spanish.

3.5 Coreference Resolution

The module of coreference resolution included in the IXA pipeline is loosely based on the Stanford Multi Sieve Pass system (Lee et al., 2013). The module takes every linguistic information it requires from the KAF layers annotated by all the previously described modules. The system consists of a number of rule-based sieves. Each sieve pass is applied in a deterministic manner, reusing the information generated by the previous sieve and the mention processing. The order in which the sieves are applied favours a highest precision approach and aims at improving the recall with the subsequent application of each of the sieve passes. This is illustrated by the evaluation results of the CoNLL 2011 Coreference Evaluation task (Lee et al., 2013), in which the Stanford's system obtained the best results.

So far we have evaluated our module on the CoNLL 2011 testset and we are a 5% behind the Stanford's system (52.8 vs 57.6 CoNLL F1), the best on that task (Lee et al., 2013). It is interesting that in our current implementation, mention-based metrics are favoured (CEAF and B^3). Still, note that these results are comparable with the results obtained by the best CoNLL 2011 participants. Currently the module performs coreference resolution only for English, although a Spanish version will be coming soon.

4 Related Work

Other NLP toolkits exist providing similar or more extensive functionalities than the IXA pipeline tools, although not many of them provide multilingual support. GATE (Cunningham, 2002) is an extensive framework supporting annotation of text. GATE has some capacity for wrapping Apache UIMA components[16], so should be able to manage distributed NLP components. However, GATE is a very large and complex system, with a corresponding steep learning curve.

Freeling (Padró and Stanilovsky, 2012) provides multilingual processing for a number of languages, incluing Spanish and English. As opposed to IXA pipeline, Freeling is a monolithic toolkit written in C++ which needs to be compiled natively. The Stanford

[11]https://github.com/morfologik/morfologik-stemming

[12]http://jwordnet.sourceforge.net/

[13]http://www.clips.ua.ac.be/conll2002/ner/

[14]http://www.clips.ua.ac.be/conll2003/ner/

[15]http://www-nlp.stanford.edu/software/lex-parser.shtml

[16]http://uima.apache.org/

CoreNLP[17] is a monolithic suite, which makes it difficult to integrate other tools in its chain.

IXA pipeline tools can easily be used piping the input with the output of another too, and it is also possible to easily replace or extend the toolchain with a third-party tool. IXA pipeline is already being used to do extensive parallel processing in the FP7 European projects OpeNER[18] and NewsReader[19].

5 Conclusion and Future Work

IXA pipeline provides a simple, efficient, accurate and ready to use set of NLP tools. Its modularity and data centric architecture makes it flexible to pick and change or integrate new linguistic annotators. Currently we offer linguistic annotation for English and Spanish, but more languages are being integrated. Furthermore, other annotations such as Semantic Role Labelling and Named Entity Disambiguation are being included in the pipeline following the same principles.

Additionally, current integrated modules are being improved: both on the quality and variety of the probabilistic models, and on specific issues such as lemmatization, and treatment of time expressions. Finally, we are adding server-mode execution into the pipeline to provide faster processing. IXA pipeline is publicly available under Apache 2.0 license: *http://adimen.si.ehu.es/web/ixa-pipes*.

Acknowledgements

TThis work has been supported by the OpeNER FP7 project under Grant No. 296451, the FP7 NewsReader project, Grant No. 316404, and by the SKATER Spanish MICINN project No TIN2012-38584-C06-01. The work of Josu Bermudez on coreference resolution is supported by a PhD Grant of the University of Deusto (http://www.deusto.es).

References

Wauter Bosma, Piek Vossen, Aitor Soroa, German Rigau, Maurizio Tesconi, Andrea Marchetti, Monica Monachini, and Carlo Aliprandi. 2009. Kaf: a generic semantic annotation format. In *Proceedings of the GL2009 Workshop on Semantic Annotation*.

X. Carreras, L. Marquez, and L. Padro. 2002. Named entity extraction using AdaBoost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4.

Stephen Clark and James Curran. 2003. Language Independent NER using a Maximum Entropy Tagger. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, pages 164–167, Edmonton, Canada.

[17]http://nlp.stanford.edu/software/corenlp.shtml

[18]http://www.opener-project.org

[19]http://www.newsreader-project.eu

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Brooke Cowan and Michael Collins. 2005. Morphology and reranking for the statistical parsing of spanish. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 795–802. Association for Computational Linguistics.

Hamish Cunningham. 2002. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

C. Fellbaum and G. Miller, editors. 1998. *Wordnet: An Electronic Lexical Database*. MIT Press, Cambridge (MA).

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370.

Jesús Giménez and Lluis Marquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*. Citeseer.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, pages 1–54, January.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May. ELRA.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, page 147155.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1-3):151–175.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. Ancora: Multilevel annotated corpora for catalan and spanish. In *LREC*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL*, pages 252–259.

XLike Project Language Analysis Services

Xavier Carreras*, **Lluís Padró***, **Lei Zhang♠**, **Achim Rettinger♠**, **Zhixing Li⋈**, **Esteban García-Cuesta◇**, **Željko Agić⋆**, **Božo Bekavac◁**, **Blaz Fortuna†**, **Tadej Štajner†**

∗ Universitat Politècnica de Catalunya, Barcelona, Spain. ◇ iSOCO S.A. Madrid, Spain.
◁ University of Zagreb, Zagreb, Croatia. ⋆ University of Potsdam, Germany.
† Jožef Stefan Institute, Ljubljana, Slovenia. ⋈ Tsinghua University, Beijing, China.
♠ Karlsruhe Institute of Technology, Karlsruhe, Germany.

Abstract

This paper presents the linguistic analysis infrastructure developed within the XLike project. The main goal of the implemented tools is to provide a set of functionalities supporting the XLike main objectives: Enabling cross-lingual services for publishers, media monitoring or developing new business intelligence applications. The services cover seven major and minor languages: English, German, Spanish, Chinese, Catalan, Slovenian, and Croatian. These analyzers are provided as web services following a lightweigth SOA architecture approach, and they are publically accessible and shared through META-SHARE.[1]

1 Introduction

Project XLike[2] goal is to develop technology able to gather documents in a variety of languages and genres (news, blogs, tweets, etc.) and to extract language-independent knowledge from them, in order to provide new and better services to publishers, media monitoring, and business intelligence. Thus, project use cases are provided by STA (Slovenian Press Agency) and Bloomberg, as well as New York Times as an associated partner.

Research partners in the project are Jožef Stefan Institute (JSI), Karlsruhe Institute of Technology (KIT), Universitat Politècnica de Catalunya (UPC), University of Zagreb (UZG), and Tsinghua University (THU). The Spanish company iSOCO is in charge of integration of all components developed in the project.

This paper deals with the language technology developed within the project XLike to convert input documents into a language-independent representation that afterwards enables knowledge aggregation.

To achieve this goal, a bench of linguistic processing pipelines is devised as the first step in the document processing flow. Then, a cross-lingual semantic annotation method, based on Wikipedia and Linked Open Data (LOD), is applied. The semantic annotation stage enriches the linguistic anaylsis with links to knowledge bases for different languages, or links to language independent representations.

2 Linguistic Analyzers

Apart from basic state-of-the-art tokenizers, lemmatizers, PoS/MSD taggers, and NE recognizers, each pipeline requires deeper processors able to build the target language-independent semantic representantion. For that, we rely on three steps: dependency parsing, semantic role labeling and word sense disambiguation. These three processes, combined with multilingual ontological resouces such as different WordNets and PredicateMatrix (López de la Calle et al., 2014), a lexical semantics resource combining WordNet, FrameNet, and VerbNet, are the key to the construction of our semantic representation.

2.1 Dependency Parsing

We use graph-based methods for dependency parsing, namely, **MSTParser**[3] (McDonald et al., 2005) is used for Chinese and Croatian, and **Treeler**[4] is used for the other languages. Treeler is a library developed by the UPC team that implements several statistical methods for tagging and parsing.

We use these tools in order to train dependency parsers for all XLike languages using standard available treebanks.

[1] *accessible and shared* here means that the services are publicly callable, not that the code is open-source. `http://www.meta-share.eu`

[2] `http://www.xlike.org`

[3] `http://sourceforge.net/projects/mstparser`

[4] `http://treeler.lsi.upc.edu`

2.2 Semantic Role Labeling

As with syntactic parsing, we are developing SRL methods with the Treeler library. In order to train models, we will use the treebanks made available by the CoNLL-2009 shared task, which provided data annotated with predicate-argument relations for English, Spanish, Catalan, German and Chinese. No treebank annotated with semantic roles exists for Slovene or Croatian. A prototype of SRL has been integrated in all pipelines (except the Slovene and Croatian pipelines). The method implemented follows a pipeline architecture described in (Lluís et al., 2013).

2.3 Word Sense Disambiguation

Word sense disambiguation is performed for all languages with a publicly available WordNet. This includes all languages in the project except Chinese. The goal of WSD is to map specific languages to a common semantic space, in this case, WN synsets. Thanks to existing connections between WN and other resources, SUMO and OpenCYC sense codes are also output when available.

Thanks to PredicateMatrix, the obtained concepts can be projected to FrameNet, achieving a normalization of the semantic roles produced by the SRL (which are treebank-dependent, and thus, not the same for all languages). The used WSD engine is the UKB (Agirre and Soroa, 2009) implementation provided by FreeLing (Padró and Stanilovsky, 2012).

2.4 Frame Extraction

The final step is to convert all the gathered linguistic information into a semantic representation. Our method is based on the notion of frame: a semantic frame is a schematic representation of a situation involving various participants. In a frame, each participant plays a role. There is a direct correspondence between roles in a frame and semantic roles; namely, frames correspond to predicates, and participants correspond to the arguments of the predicate. We distinguish three types of participants: entities, words, and frames.

Entities are nodes in the graph connected to real-world entities as described in Section 3. Words are common words or concepts, linked to general ontologies such as WordNet. Frames correspond to events or predicates described in the document. Figure 1 shows an example sentence, the extracted frames and their arguments.

It is important to note that frames are a more general representation than SVO-triples. While SVO-triples represent a binary relation between two participants, frames can represent n-ary relations (e.g. predicates with more than two arguments, or with adjuncts). Frames also allow representing the sentences where one of the arguments is in turn a frame (as is the case with *plan to make* in the example).

Finally, although frames are extracted at sentence level, the resulting graphs are aggregated in a single semantic graph representing the whole document via a very simple coreference resolution based on detecting named entity aliases and repetitions of common nouns. Future improvements include using an state-of-the-art coreference resolution module for languages where it is available.

3 Cross-lingual Semantic Annotation

This step adds further semantic annotations on top of the results obtained by linguistic processing. All XLike languages are covered. The goal is to map word phrases in different languages into the same semantic interlingua, which consists of resources specified in knowledge bases such as Wikipedia and Linked Open Data (LOD) sources. Cross-lingual semantic annotation is performed in two stages: (1) first, candidate concepts in the knowledge base are linked to the linguistic resources based on a newly developed cross-lingual linked data lexica, called xLiD-Lexica, (2) next the candidate concepts get disambiguated based on the personalized PageRank algorithm by utilizing the structure of information contained in the knowledge base.

The xLiD-Lexica is stored in RDF format and contains about 300 million triples of cross-lingual groundings. It is extracted from Wikipedia dumps of July 2013 in English, German, Spanish, Catalan, Slovenian and Chinese, and based on the canonicalized datasets of DBpedia 3.8 containing triples extracted from the respective Wikipedia whose subject and object resource have an equivalent English article.

4 Web Service Architecture Approach

The different language functionalities are implemented following the service oriented architecture (SOA) approach defined in the project XLike. Therefore all the pipelines (one for each language) have been implemented as web services and may

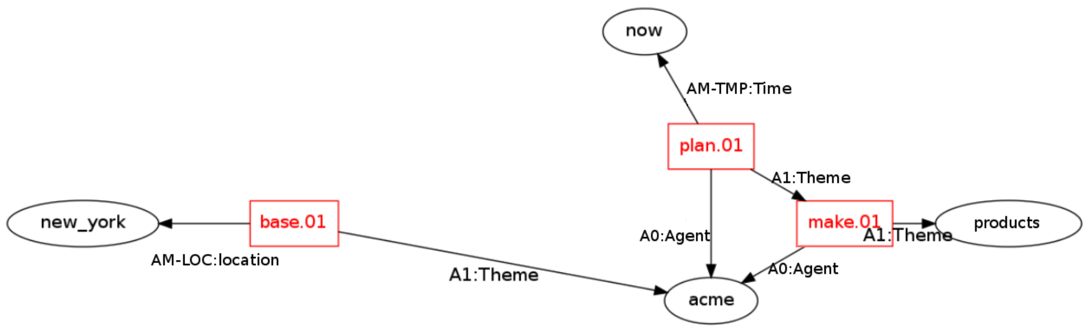


Figure 1: Graphical representation of frames in the sentence *Acme, based in New York, now plans to make computer and electronic products.*

be requested to produce different levels of analysis (e.g. tokenization, lemmatization, NERC, parsing, relation extraction). This approach is very appealing due to the fact that it allows to treat every language independently and execute the whole language analysis process at different threads or computers allowing an easier parallelization (e.g. using external high perfomance platforms such as Amazon Elastic Compute Cloud EC2[5]) as needed. Furthermore it also provides independent development lifecycles for each language which is crucial in this type of research projects. Recall that these web services can be deployed locally or remotely, maintaining the option of using them in a standalone configuration.

The main structure for each one of the pipelines is described below:

- **Spanish**, **English**, and **Catalan**: all modules are based on FreeLing (Padró and Stanilovsky, 2012) and Treeler.
- **German**: German shallow processing is based on OpenNLP[6], Stanford POS tagger and NE extractor (Toutanova et al., 2003; Finkel et al., 2005). Dependency parsing, semantic role labeling, word sense disambiguation, and SRL-based frame extraction are based on FreeLing and Treeler.
- **Slovene**: Slovene shallow processing is provided by JSI Enrycher[7] (Štajner et al., 2010), which consists of the Obeliks morphosyntactic analysis library (Grčar et al., 2012), the LemmaGen lemmatizer (Juršič et al., 2010) and a CRF-based entity extractor (Štajner et al., 2012). Dependency parsing, word sense disambiguation are based on FreeLing and Treeler. Frame extraction is rule-based since no SRL corpus is available for Slovene.
- **Croatian**: Croatian shallow processing is based on proprietary tokenizer, POS/MSD-tagging and lemmatisaton system (Agić et al., 2008), NERC system (Bekavac and Tadić, 2007) and dependency parser (Agić, 2012). Word sense disambiguation is based on FreeLing. Frame extraction is rule-based since no SRL corpus is available for Croatian.
- **Chinese**: Chinese shallow and deep processing is based on a word segmentation component ICTCLAS[8] and a semantic dependency parser trained on CSDN corpus. Then, rule-based frame extraction is performed (no SRL corpus nor WordNet are available for Chinese).

Each language analysis service is able to process thousands of words per second when performing shallow analysis (up to NE recognition), and hundreds of words per second when producing the semantic representation based on full analysis. Moreover, the web service architecture enables the same server to run a different thread for each client, thus taking advantage of multiprocessor capabilities.

The components of the cross-lingual semantic annotation stage are:

- **xLiD-Lexica**: The cross-lingual groundings in xLiD-Lexica are translated into RDF data and are accessible through a SPARQL endpoint, based on OpenLink Virtuoso[9] as the back-end database engine.

[5] http://aws.amazon.com/ec2/
[6] http://opennlp.apache.org
[7] http://enrycher.ijs.si
[8] http://ictclas.org/
[9] http://virtuoso.openlinksw.com/

- **Semantic Annotation**: The cross-lingual semantic annotation service is based on the xLiD-Lexica for entity mention recognition and the JUNG Framework[10] for graph-based disambiguation.

5 Conclusion

We presented the web service based architecture used in XLike FP7 project to linguistically analyze large amounts of documents in seven different languages. The analysis pipelines perform basic processing as tokenization, PoS-tagging, and named entity extraction, as well as deeper analysis such as dependency parsing, word sense disambiguation, and semantic role labelling. The result of these linguistic analyzers is a semantic graph capturing the main events described in the document and their core participants.

On top of that, the cross-lingual semantic annotation component links the resulting linguistic resources in one language to resources in a knowledge bases in any other language or to language independent representations. This semantic representation is later used in XLike for document mining purposes such as enabling cross-lingual services for publishers, media monitoring or developing new business intelligence applications.

The described analysis services are currently available via META-SHARE as callable RESTful services.

Acknowledgments

This work was funded by the European Union through project XLike (FP7-ICT-2011-288342).

References

Željko Agić, Marko Tadić, and Zdravko Dovedan. 2008. Improving part-of-speech tagging accuracy for Croatian by morphological analysis. *Informatica*, 32(4):445–451.

Željko Agić. 2012. K-best spanning tree dependency parsing with verb valency lexicon reranking. In *Proceedings of COLING 2012: Posters*, pages 1–12, Mumbai, India, December. The COLING 2012 Organizing Committee.

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL-2009)*, Athens, Greece.

Božo Bekavac and Marko Tadić. 2007. Implementation of Croatian NERC system. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing (BSNLP2007), Special Theme: Information Extraction and Enabling Technologies*, pages 11–18. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL'05)*, pages 363–370.

Miha Grčar, Simon Krek, and Kaja Dobrovoljc. 2012. Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. In *Zbornik Osme konference Jezikovne tehnologije*, Ljubljana, Slovenia.

Matjaz Juršič, Igor Mozetič, Tomaz Erjavec, and Nada Lavrač. 2010. Lemmagen: Multilingual lemmatisation with induced ripple-down rules. *Journal of Universal Computer Science*, 16(9):1190–1214.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational Linguistics*, 1:219–230.

Maddalen López de la Calle, Egoitz Laparra, and German Rigau. 2014. First steps towards a predicate matrix. In *Proceedings of the Global WordNet Conference (GWC 2014)*, Tartu, Estonia, January. GWA.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, June.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May. ELRA.

Tadej Štajner, Delia Rusu, Lorand Dali, Blaž Fortuna, Dunja Mladenić, and Marko Grobelnik. 2010. A service oriented framework for natural language text enrichment. *Informatica*, 34(3):307–313.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Lin- guistics on Human Language Technology (NAACL'03)*.

Tadej Štajner, Tomaž Erjavec, and Simon Krek. 2012. Razpoznavanje imenskih entitet v slovenskem besedilu. In *In Proceedings of 15th Internation Multiconference on Information Society - Jezikovne Tehnologije*, Ljubljana, Slovenia.

[10]Java Universal Network/Graph Framework `http://jung.sourceforge.net/`

Semantic Annotation, Analysis and Comparison: A Multilingual and Cross-lingual Text Analytics Toolkit

Lei Zhang
Institute AIFB
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
`l.zhang@kit.edu`

Achim Rettinger
Institute AIFB
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
`rettinger@kit.edu`

Abstract

Within the context of globalization, multilinguality and cross-linguality for information access have emerged as issues of major interest. In order to achieve the goal that users from all countries have access to the same information, there is an impending need for systems that can help in overcoming language barriers by facilitating multilingual and cross-lingual access to data. In this paper, we demonstrate such a toolkit, which supports both service-oriented and user-oriented interfaces for semantically annotating, analyzing and comparing multilingual texts across the boundaries of languages. We conducted an extensive user study that shows that our toolkit allows users to solve cross-lingual entity tracking and article matching tasks more efficiently and with higher accuracy compared to the baseline approach.

1 Introduction

Automatic text understanding has been an unsolved research problem for many years. This partially results from the dynamic and diverging nature of human languages, which results in many different varieties of natural language. These variations range from the individual level, to regional and social dialects, and up to seemingly separate languages and language families. In recent years there have been considerable achievements in approaches to computational linguistics exploiting the information across languages. This progress in multilingual and cross-lingual text analytics is largely due to the increased availability of multilingual knowledge bases such as Wikipedia, which helps at scaling the traditionally monolingual tasks to multilingual and cross-lingual applications. From the application side, there is a clear need for multilingual and cross-lingual text analytics technologies and services.

Text analytics in this work is defined as three tasks: (i) *semantic annotation* by linking entity mentions in the documents to their corresponding representations in the knowledge base; (ii) *semantic analysis* by linking the documents by topics to the relevant resources in the knowledge base; (iii) *semantic comparison* by measuring semantic relatedness between documents. While *multilingual* text analytics addresses these tasks for multiple languages, *cross-lingual* text analytics goes one step beyond, as it faces these tasks across the boundaries of languages, i.e., the text to be processed and the resources in the knowledge base, or the documents to be compared, are in different languages.

Due to the ever growing richness of its content, Wikipedia has been increasingly gaining attention as a precious knowledge base that contains an enormous number of entities and topics in diverse domains. In addition, Wikipedia pages that provide information about the same concept in different languages are connected through cross-language links. Therefore, we use Wikipedia as the central knowledge base.

With the goal of overcoming language barriers, we would like to demonstrate our multilingual and cross-lingual text analytics toolkit, which supports both service-oriented and user-oriented interfaces for semantically annotating, analyzing and comparing multilingual texts across the boundaries of languages.

2 Techniques

In this section, we first present the techniques behind our toolkit w.r.t. its three components: semantic annotation (Sec. 2.1), semantic analysis and semantic comparison (Sec. 2.2).

2.1 Wikipedia-based Annotation

The process of augmenting phrases in text with links to their corresponding Wikipedia articles (in the sense of Wikipedia-based annotation) is known as *wikification*. There is a large body of work that links phrases in unstructured text to relevant Wikipedia articles. While Mihalcea and Csomai (Mihalcea and Csomai, 2007) met the challenge of wikification by using link probabilities obtained from Wikipedia's articles and by a comparison of features extracted from the context of the phrases, Milne and Witten (Milne and Witten, 2008) could improve the wikification service significantly by viewing wikification even more as a supervised machine learning task: Wikipedia is used here not only as a source of information to point to, but also as training data to find always the appropriate link.

For multilingual semantic annotation, we adopted the wikification system in (Milne and Witten, 2008) and trained it for each language using the corresponding Wikipedia version. To perform cross-lingual semantic annotation, we extended the wikification system by making use of the cross-language links in Wikipedia to find the corresponding Wikipedia articles in the different target languages. More details can be found in our previous work (Zhang et al., 2013).

2.2 Explicit Semantic Analysis

Explicit Semantic Analysis (ESA) has been proposed as an approach for semantic modeling of natural language text (Gabrilovich and Markovitch, 2006). Based on a given set of concepts with textual descriptions, ESA defines the concept-based representation of documents. Various sources for concept definitions have been used, such as Wikipedia and Reuters Corpus. Using the concept-based document representation, ESA has been successfully applied to compute semantic relatedness between texts (Gabrilovich and Markovitch, 2007). In the context of the cross-language information retrieval (CLIR) task, ESA has been extended to a cross-lingual setting (CL-ESA) by mapping the semantic document representation from a concept space of one language to an interlingual concept space (Sorg and Cimiano, 2008).

The semantic analysis and semantic comparison components of our toolkit are based on CL-ESA in (Sorg and Cimiano, 2008). The semantic analysis component takes as input a document in a source language and maps it to a high-dimensional vector in the interlingual concept space, such that each dimension corresponds to an Wikipedia article in any target language acting as a concept. For semantic comparison, the documents in different languages are first translated into vectors in the interlingual concept space and then the cross-lingual semantic relatedness between the documents in different languages can be calculated using the standard similarity measure between the resulting vectors.

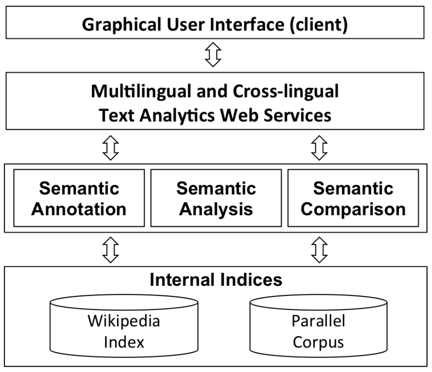


Figure 2: Architecture of our Toolkit.

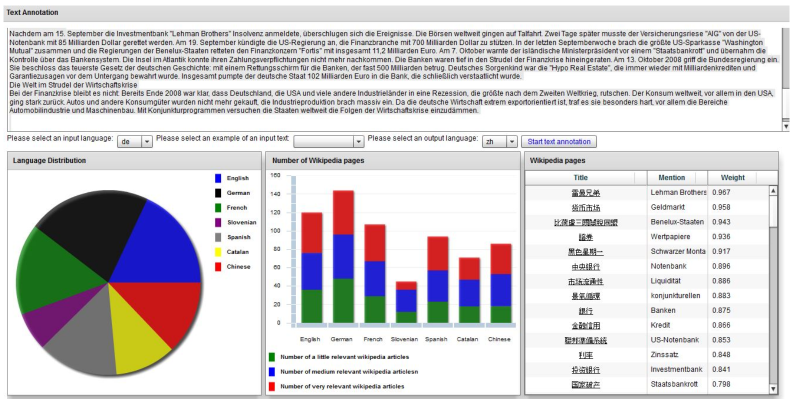
3 Implementation

Our multilingual and cross-lingual toolkit is implemented using a client-server architecture with communication over HTTP using a XML schema defined in XLike project[1]. The server is a RESTful web service and the client user interface is implemented using Adobe Flex as both Desktop and Web Applications. The toolkit can easily be extended or adapted to switch out the server or client. In this way, it supports both service-oriented and user-oriented interfaces for semantically annotating, analyzing and comparing multilingual texts across the boundaries of languages. The architecture of our toolkit is shown in Figure 2.

For all three components, namely semantic annotation, analysis and comparison, we use Wikipedia as the central knowledge base. Table 1 shows the statistics of the Wikipedia articles in English, German, Spanish and French as well as the cross-language links between the them in these languages extracted from Wikipedia snapshots of May 2012[2], which are used to build our toolkit.

We now describe the user interfaces of these

[1]http://www.xlike.org/
[2]http://dumps.wikimedia.org/


Text Annotation

Nachdem am 15. September die Investmentbank "Lehman Brothers" Insolvenz anmeldete, überschlugen sich die Ereignisse. Die Börsen weltweit gingen auf Talfahrt. Zwei Tage später musste der Versicherungsriese "AIG" von der US-Notenbank mit 85 Milliarden Dollar gerettet werden. Am 19. September kündigte die US-Regierung an, die Finanzbranche mit 700 Milliarden Dollar zu stützen. In der letzten Septemberwoche brach die größte US-Sparkasse "Washington Mutual" zusammen und die Regierungen der Benelux-Staaten retteten den Finanzkonzern "Fortis" mit insgesamt 11,2 Milliarden Euro. Am 7. Oktober warnte der isländische Ministerpräsident vor einem "Staatsbankrott" und übernahm die Kontrolle über das Bankensystem. Die Insel im Atlantik konnte ihren Zahlungsverpflichtungen nicht mehr nachkommen. Die Banken waren tief in den Strudel der Finanzkrise hineingeraten. Am 13. Oktober 2008 griff die Bundesregierung ein. Sie beschloss das teuerste Gesetz der deutschen Geschichte: mit einem Rettungsschirm für die Banken, der fast 500 Milliarden betrug. Deutsches Sorgenkind war die "Hypo Real Estate", die immer wieder mit Milliardenkrediten und Garantiezusagen vor dem Untergang bewahrt wurde. Insgesamt pumpte der deutsche Staat 102 Milliarden Euro in die Bank, die schließlich verstaatlicht wurde.
Die Welt im Strudel der Wirtschaftskrise
Bei der Finanzkrise bleibt es nicht: Bereits Ende 2008 war klar, dass Deutschland, die USA und viele andere Industrieländer in eine Rezession, die größte nach dem Zweiten Weltkrieg, rutschen. Der Konsum weltweit, vor allem in den USA, ging stark zurück. Autos und andere Konsumgüter wurden nicht mehr gekauft, die Industrieproduktion brach massiv ein. Da die deutsche Wirtschaft extrem exportorientiert ist, traf es sie besonders hart, vor allem die Bereiche Automobilindustrie und Maschinenbau. Mit Konjunkturprogrammen versuchen die Staaten weltweit die Folgen der Wirtschaftskrise einzudämmen.

Please select an input language: de
Please select an example of an input text:
Please select an output language: zh

Wikipedia pages

Title	Mention	Weight
雷曼兄弟	Lehman Brothers	0.967
货币市场	Geldmarkt	0.958
比荷盧三國關稅同盟	Benelux-Staaten	0.943
證券	Wertpapiere	0.936
黑色星期一	Schwarzer Monta	0.917
中央银行	Notenbank	0.896
市场流通性	Liquidität	0.886
景氣循環	konjunkturellen	0.883
銀行	Banken	0.875
金融信用	Kredit	0.866
聯邦準備系統	US-Notenbank	0.853
利率	Zinssatz	0.848
投资银行	Investmentbank	0.841
国家破产	Staatsbankrott	0.798

Figure 1: Screenshot of the Semantic Annotation Component of our Toolkit.

	English (EN)	German (DE)	Spanish (ES)	French (FR)
#Articles	4,014,643	1,438,325	896,691	1,234,567

(a) Number of articles.

	EN-DE	EN-ES	EN-FR	DE-ES	DE-FR	ES-FR
#Links (→)	721,878	568,210	779,363	295,415	455,829	378,052
#Links (←)	718,401	581,978	777,798	302,502	457,306	370,552
#Links (merged)	722,069	593,571	795,340	307,130	464,628	383,851

(b) Number of cross-language links.

Table 1: Statistics about Wikipedia.

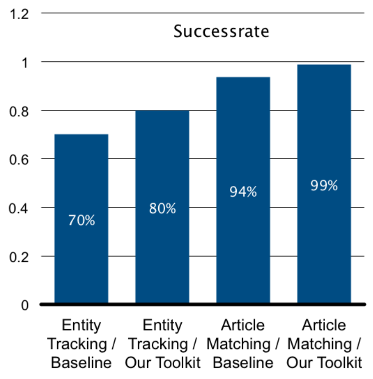
components. Due to the lack of space, we only show the screenshot of the semantic annotation component in Figure 1. The semantic annotation component allows the users to find the entities in Wikipedia mentioned in the input document. Given the input document in one language, the users can select the output language, namely the language of Wikipedia articles describing the mentioned entities. In the left pie chart, the users can see the percentage of Wikipedia articles in different languages as annotations of the input document. According to their weights, the Wikipedia articles in each language are organized in 3 relevance categories: high, medium and low. In the middle bar chart, the number of Wikipedia articles in each language and in each category is illustrated. The right data grid provides the Wikipedia article titles with their weights in the output language and the mentions in the input document. Clicking an individual title opens the corresponding Wikipedia article in the output language. The semantic analysis component has the similar user interface as the semantic annotation component. The difference is that the Wikipedia articles listed in the right data grid are topically relevant to the input documents instead of being mentioned as entities. Regarding the user interface of semantic comparison component, the main inputs are two documents that might be in different languages and the output is the semantic relatedness between them.

4 User Study

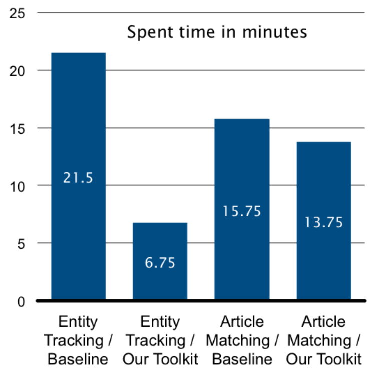
We conducted a task-based user study and the goal is to assess the effectiveness and usability of our multilingual and cross-lingual text analytics toolkit. We design two tasks reflecting the real-life information needs, namely *entity tracking* and *article matching*, to assess the functionality of our toolkit from different perspectives. The *entity tracking* task is to detect mentions of the given entities in the articles, where the descriptions of the entities and the articles are in different languages. Given articles in one language as context, the *article matching* task is to find the most similar articles in another language.

The participants of our user study are 16 volunteers and each of them got both tasks, which they had to solve in two ways: (1) using a major online machine translation service as baseline and (2) using our multilingual and cross-lingual text analytics toolkit with all the functionality. For each task, we randomly selected 10 parallel articles in English, French and Spanish from the JRC-Acquis parallel corpus[3]. After a survey,

[3]`http://langtech.jrc.it/JRC-Acquis.html`



(a) Avg. successrate per task / method



(b) Avg. time spent per task / method

Figure 3: Evaluation Results of the User Study.

we decided to provide the entity descriptions for entity tracking task and the context documents for article matching task in English, which all participants can speak. Regarding the articles to be processed, we set up the tasks using Spanish articles for the participants who do not know Spanish, and tasks with French articles for the participants who cannot speak French.

To measure the overall effectiveness of our toolkit, we have analysed the ratio of tasks that were completed successfully and correctly and the time the participants required for the tasks. The average success rate and time spent per task and per method are illustrated in Figure 3. For entity tracking task, we observe that a success rate of 80% was achieved using our toolkit in comparison with the success rate of 70% yielded by using the baseline. In addition, there is a significant gap between the time spent using different methods. While it took 21.5 minutes on average to solve the task using the baseline, only 6.75 minutes were needed when using our toolkit. Regarding the article matching task, both methods performed very well. Using our toolkit obtained a slightly higher success rate of 99% than 94% using the baseline. The time spent using both methods is not so different. The participants spent 15.75 minutes on average using the baseline while 2 minutes less were needed using our toolkit.

In terms of the user study, our toolkit is more effective than the baseline for both entity tracking and article matching tasks. Therefore, we conclude that our toolkit provides useful functionality to make searching entities, analyzing and comparing articles more easily and accurately in the multilingual and cross-lingual scenarios.

Acknowledgments

This work was supported by the European Community's Seventh Framework Programme FP7-ICT-2011-7 (XLike, Grant 288342).

References

[Gabrilovich and Markovitch2006] Evgeniy Gabrilovich and Shaul Markovitch. 2006. Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *AAAI*, pages 1301–1306.

[Gabrilovich and Markovitch2007] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on artificial intelligence*, volume 6, page 12.

[Mihalcea and Csomai2007] Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *In CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM.

[Milne and Witten2008] David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA. ACM.

[Sorg and Cimiano2008] P. Sorg and P. Cimiano. 2008. Cross-lingual Information Retrieval with Explicit Semantic Analysis. In *Working Notes of the Annual CLEF Meeting*.

[Zhang et al.2013] Lei Zhang, Achim Rettinger, Michael Frber, and Marko Tadic. 2013. A comparative evaluation of cross-lingual text annotation techniques. In *CLEF*, pages 124–135.

RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger

Dat Quoc Nguyen[1] and **Dai Quoc Nguyen**[1] and **Dang Duc Pham**[2] and **Son Bao Pham**[1]
[1] Faculty of Information Technology
University of Engineering and Technology
Vietnam National University, Hanoi
{datnq, dainq, sonpb}@vnu.edu.vn
[2] L3S Research Center, Germany
pham@L3S.de

Abstract

This paper describes our robust, easy-to-use and language independent toolkit namely RDRPOSTagger which employs an error-driven approach to automatically construct a Single Classification Ripple Down Rules tree of transformation rules for POS tagging task. During the demonstration session, we will run the tagger on data sets in 15 different languages.

1 Introduction

As one of the most important tasks in Natural Language Processing, Part-of-speech (POS) tagging is to assign a tag representing its lexical category to each word in a text. Recently, POS taggers employing machine learning techniques are still mainstream toolkits obtaining state-of-the-art performances[1]. However, most of them are time-consuming in learning process and require a powerful computer for possibly training machine learning models.

Turning to rule-based approaches, the most well-known method is proposed by Brill (1995). He proposed an approach to automatically learn transformation rules for the POS tagging problem. In the Brill's tagger, a new selected rule is learned on a context that is generated by all previous rules, where a following rule will modify the outputs of all the preceding rules. Hence, this procedure returns a difficulty to control the interactions among a large number of rules.

Our RDRPOSTagger is presented to overcome the problems mentioned above. The RDRPOSTagger exploits a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules (SCRDR) tree (Richards, 2009). It accepts interactions between rules, but a rule only changes the outputs of some previous rules in a controlled context. All rules are structured in a SCRDR tree which allows a new exception rule to be added when the tree returns an incorrect classification. A specific description of our new RDRPOSTagger approach is detailed in (Nguyen et al., 2011).

Packaged in a 0.6MB zip file, implementations in Python and Java can be found at the tagger's website *http://rdrpostagger.sourceforge.net/*. The following items exhibit properties of the tagger:

• The RDRPOSTagger is easy to configure and train. There are only two threshold parameters utilized to learn the rule-based model. Besides, the tagger is very simple to use with standard input and output, having clear usage and instructions available on its website.

• The RDRPOSTagger is language independent. This POS tagging toolkit has been successfully applied to English and Vietnamese. To train the toolkit for other languages, users just provide a lexicon of words and the most frequent associated tags. Moreover, it can be easily combined with existing POS taggers to reach an even better result.

• The RDRPOSTagger obtains very competitive accuracies. On Penn WSJ Treebank corpus (Marcus et al., 1993), taking WSJ sections 0-18 as the training set, the tagger achieves a competitive performance compared to other state-of-the-art English POS taggers on the test set of WSJ sections 22-24. For Vietnamese, it outperforms all previous machine learning-based POS tagging systems to obtain an up-to-date highest result on the Vietnamese Treebank corpus (Nguyen et al., 2009).

• The RDRPOSTagger is fast. For instance in English, the time[2] taken to train the tagger on the WSJ sections 0-18 is **40** minutes. The tagging speed on the test set of the WSJ sections 22-24 is **2800** words/second accounted for the latest implementation in Python whilst it is **92k** words/second

[1] http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

[2] Training and tagging times are computed on a Windows-7 OS computer of Core 2Duo 2.4GHz & 3GB of memory.

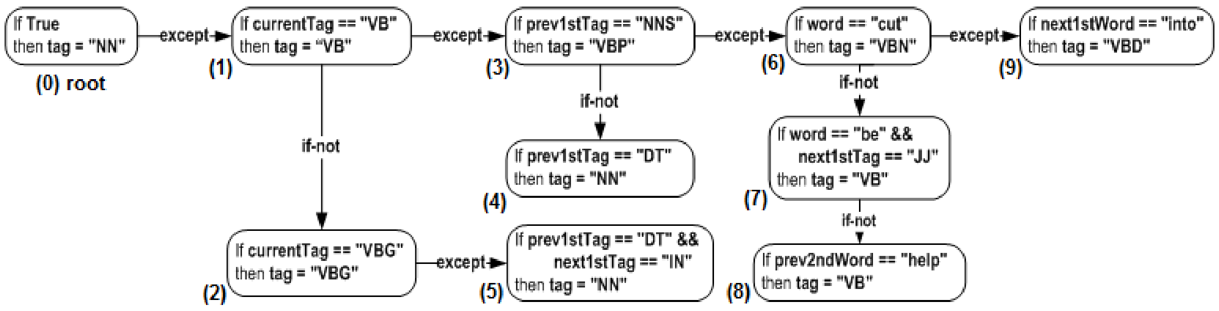


Figure 1: A part of our SCRDR tree for English POS tagging.

for the implementation in Java.

2 SCRDR methodology

A SCRDR tree (Richards, 2009) is a binary tree with two distinct types of edges. These edges are typically called *except* and *if-not* edges. Associated with each node in a tree is a *rule*. A rule has the form: *if* α *then* β where α is called the *condition* and β is referred to as the *conclusion*.

Cases in SCRDR are evaluated by passing a case to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied by the case, the case is passed on to the exception child of N using the *except* link if it exists. Otherwise, the case is passed on to the N's *if-not* child. The conclusion given by this process is the conclusion from the last node in the SCRDR tree which *fired* (satisfied by the case). To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default* node.

A new node containing a new rule (i.e. a new exception rule) is added to an SCRDR tree when the evaluation process returns the *wrong* conclusion. The new node is attached to the last node in the evaluation path of the given case with the *except* link if the last node is the *fired* one. Otherwise, it is attached with the *if-not* link.

For example with the SCRDR tree in the figure 1, given a case *"as/IN investors/NNS anticipate/VB a/DT recovery/NN"* where *"anticipate/VB"* is the current word and tag pair, the case satisfies the conditions of the rules at nodes (0), (1) and (3), it then is passed to the node (6) (utilizing except links). As the case does not satisfy the condition of the rule at node (6), it will be transferred to node (7) using if-not link. Since the case does not fulfill the conditions of the rules at nodes (7) and (8), we have the evaluation path (0)-(1)-(3)-(6)-(7)-(8) with fired node (3). Therefore, the tag for *"anticipate"* is concluded as "VBP".

Rule (1) - the rule at node (1) - is the exception rule[3] of the default rule (0). As node (2) is the if-not child node of the node (1), the associated rule (2) is also an exception rule of the rule (0). Similarly, both rules (3) and (4) are exception rules of the rule (1) whereas all rules (6), (7) and (8) are exception rules of the rule (3), and so on. Thus, the exception structure of the SCRDR tree extends to 4 levels: rules (1) and (2) at layer 1, rules (3), (4) and (5) at layer 2, rules (6), (7) and (8) at layer 3, and rule (9) at layer 4.

3 The RDRPOSTagger toolkit

The toolkit consists of four main components: Utility, Initial-tagger, SCRDR-learner and SCRDR-tagger.

3.1 The Utility

The major functions of this component are to evaluate tagging performances (displaying accuracy results), and to create a lexicon of words and the most frequent associated tags as well as to extract *Raw corpus* from an input golden training corpus.

3.2 The Initial-tagger

The initial-tagger developed in the RDRPOSTagger toolkit is based on the lexicon which is generated in the use of the Utility component to assign a tag for each word. To deal with unknown words, the initial-tagger utilizes several regular expressions or heuristics for English and Vietnamese whereas the most frequent tag in the training corpus is exploited to label unknown-words when adapting to other languages.

3.3 The SCRDR-learner

The SCRDR-learner component uses a failure-driven method to automatically build a SCRDR tree of transformation rules. Figure 3 describes the learning process of the learner.

[3]The default rule is the unique rule which is not an exception rule of any other rule. Every rule in layer n is an exception rule of a rule in layer $n-1$.

#12: *if* next1stTag == **"object.next1stTag"** *then* tag = **"correctTag"**
#14: *if* prev1stTag == **"object.prev1stTag"** *then* tag = **"correctTag"**
#18: *if* word == **"object.word"** && next1stTag == **"object.next1stTag"** *then* tag = **"correctTag"**

Figure 2: Rule template examples.

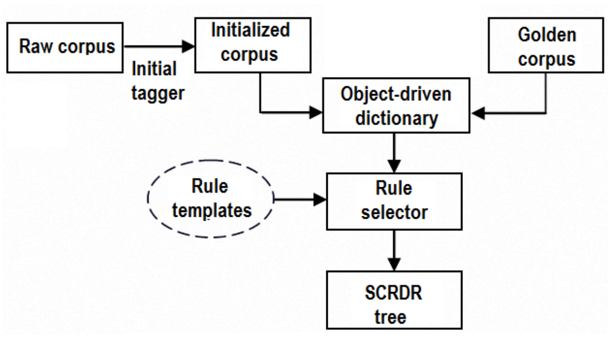


Figure 3: The diagram of the learning process of the learner.

The *Initialized corpus* is returned by performing the Initial-tagger on the Raw corpus. By comparing the initialized one with the *Golden corpus*, an *Object-driven dictionary* of pairs (***O**bject, correctTag*) is produced in which *Object* captures the 5-word window context covering the current word and its tag in following format (*previous* 2^{nd} *word / previous* 2^{nd} *tag, previous* 1^{st} *word / previous* 1^{st} *tag, word / currentTag, next* 1^{st} *word / next* 1^{st} *tag, next* 2^{nd} *word / next* 2^{nd} *tag*) from the initialized corpus, and the *correctTag* is the corresponding tag of the current word in the golden corpus.

There are 27 *Rule templates* applied for *Rule selector* which is to select the most suitable rules to build the *SCRDR tree*. Examples of the rule templates are shown in figure 2 where elements in bold will be replaced by concrete values from *Object*s in the object-driven dictionary to create concrete rules. The SCRDR tree of rules is initialized by building the default rule and all exception rules of the default one in form of *if currentTag = "**TAG**" then tag = "**TAG**"* at the layer-1 exception structure, for example rules (1) and (2) in the figure 1, and the like. The learning approach to construct new exception rules to the tree is as follows:

• At a node-F in the SCRDR tree, let SO be the set of Objects from the object-driven dictionary, which those Objects are fired at the node-F but their initialized tags are incorrect (the *currentTag* is not the *correctTag* associated). It means that node-F gives wrong conclusions to all Objects in the SO set.

• In order to select a new exception rule of the rule at node-F from all concrete rules which are generated for all Objects in the SO set, the selected rule have to satisfy constraints: (i) The rule must be unsatisfied by cases for which node-F has already given correct conclusions. This constraint does not apply to node-F at layer-1 exception structure. (ii) The rule must associate to a highest score value of subtracting B from A in comparison to other ones, where A and B are the numbers of the SO's Objects which are correctly and incorrectly concluded by the rule respectively. (iii) And the highest value is not smaller than a given threshold.

The SCRDR-learner applies two threshold parameters: first threshold is to choose exception rules at the layer-2 exception structure (e.g rules (3), (4) and (5) in figure 1), and second threshold is to select rules for higher exception layers.

• The process to add new exception rules is repeated until there is no rule satisfying the constraints above. At each iteration, a new rule is added to the current SCRDR tree to correct error conclusions made by the tree.

3.4 The SCRDR-tagger

The SCRDR-tagger component is to perform the POS tagging on a raw text corpus where each line is a sequence of words separated by white space characters. The component labels the text corpus by using the Initial-tagger. It slides due to a left-to-right direction on a 5-word window context to generate a corresponding Object for each initially tagged word. The Object is then classified by the learned SCRDR tree model to produce final conclusion tag of the word as illustrated in the example in the section 2.

4 Evaluation

The RDRPOSTagger has already been successfully applied to English and Vietnamese corpora.

4.1 Results for English

Experiments for English employed the Penn WSJ Treebank corpus to exploit the WSJ sections 0-18 (38219 sentences) for training, the WSJ sections 19-21 (5527 sentences) for validation and the WSJ sections 22-24 (5462 sentences) for test.

Using a lexicon created in the use of the training set, the Initial-tagger obtains an accuracy of 93.51% on the test set. By varying the thresholds on the validation set, we have found the most suitable values[4] of 3 and 2 to be used for evaluating the RDRPOSTagger on the test set. Those thresholds return a SCRDR tree model of 2319 rules in a 4-level exception structure. The training time and tagging speed for those thresholds are mentioned in the introduction section. On the same test set, the RDRPOSTagger achieves a performance at 96.49% against 96.46% accounted for the state-of-the-art POS tagger TnT (Brants, 2000).

For another experiment, only in training process: 1-time occurrence words in training set are initially tagged as out-of-dictionary words. With a learned tree model of 2418 rules, the tagger reaches an accuracy of 96.51% on the test set.

Retraining the tagger utilizing another initial tagger[5] developed in the Brill's tagger (Brill, 1995) instead of the lexicon-based initial one, the RDRPOSTagger gains an accuracy result of 96.57% which is slightly higher than the performance at 96.53% of the Brill's.

4.2 Results for Vietnamese

In the first Evaluation Campaign[6] on Vietnamese Language Processing, the POS tagging track provided a golden training corpus of 28k sentences (631k words) collected from two sources of the national VLSP project and the Vietnam Lexicography Center, and a raw test corpus of 2100 sentences (66k words). The training process returned a SCRDR tree of 2896 rules[7]. Obtaining a highest performance on the test set, the RDRPOSTagger surpassed all other participating systems.

We also carry out POS tagging experiments on the golden corpus of 28k sentences and on the Vietnamese Treebank of 10k sentences (Nguyen et al., 2009) according to 5-fold cross-validation scheme[8]. The average accuracy results are presented in the table 1. Achieving an accuracy of 92.59% on the Vietnamese Treebank, the RDRPOSTagger outperforms previous Maximum Entropy Model, Conditional Random Field and Support Vector Machine-based POS tagging systems (Tran et al., 2009) on the same evaluation scheme.

Table 1: Accuracy results for Vietnamese

Corpus	Initial-tagger	RDRPOSTagger
28k	91.18%	93.42%
10k	90.59%	92.59%

5 Demonstration and Conclusion

In addition to English and Vietnamese, in the demonstration session, we will present promising experimental results and run the RDRPOSTagger for other languages including Bulgarian, Czech, Danish, Dutch, French, German, Hindi, Italian, Lao, Portuguese, Spanish, Swedish and Thai. We will also let the audiences to contribute their own data sets for retraining and testing the tagger.

In this paper, we describe the rule-based POS tagging toolkit RDRPOSTagger to automatically construct transformation rules in form of the SCRDR exception structure. We believe that our robust, easy-to-use and language-independent toolkit RDRPOSTagger can be useful for NLP/CL-related tasks.

[4]The thresholds 3 and 2 are reused for all other experiments in English and Vietnamese.

[5]The initial tagger gets a result of 93.58% on the test set.

[6]http://uet.vnu.edu.vn/rivf2013/campaign.html

[7]It took 100 minutes to construct the tree leading to tagging speeds of 1100 words/second and 45k words/second for the implementations in Python and Java, respectively, on the computer of Core 2Duo 2.4GHz & 3GB of memory.

[8]In each cross-validation run, one fold is selected as test set, 4 remaining folds are merged as training set. The initial tagger exploits a lexicon generated from the training set. In training process, 1-time occurrence words are initially labeled as out-of-lexicon words.

References

Thorsten Brants. 2000. TnT: a statistical part-of-speech tagger. In *Proc. of 6th Applied Natural Language Processing Conference*, pages 224–231.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguist.*, 21(4):543–565.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, 19(2):313–330.

Phuong Thai Nguyen, Xuan Luong Vu, Thi Minh Huyen Nguyen, Van Hiep Nguyen, and Hong Phuong Le. 2009. Building a Large Syntactically-Annotated Corpus of Vietnamese. In *Proc. of LAW III workshop*, pages 182–185.

Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and Dang Duc Pham. 2011. Ripple Down Rules for Part-of-Speech Tagging. In *Proc. of 12th CICLing - Volume Part I*, pages 190–201.

Debbie Richards. 2009. Two decades of ripple down rules research. *Knowledge Engineering Review*, 24(2):159–184.

Oanh Thi Tran, Cuong Anh Le, Thuy Quang Ha, and Quynh Hoang Le. 2009. An experimental study on vietnamese pos tagging. *Proc. of the 2009 International Conference on Asian Language Processing*, pages 23–27.

Morfessor 2.0: Toolkit for statistical morphological segmentation

Peter Smit[1]
peter.smit@aalto.fi

Sami Virpioja[2]
sami.virpioja@aalto.fi

Stig-Arne Grönroos[1]
stig-arne.gronroos@aalto.fi

Mikko Kurimo[1]
mikko.kurimo@aalto.fi

[1]Department of Signal Processing and Acoustics, Aalto University
[2]Department of Information and Computer Science, Aalto University

Abstract

Morfessor is a family of probabilistic machine learning methods for finding the morphological segmentation from raw text data. Recent developments include the development of semi-supervised methods for utilizing annotated data. Morfessor 2.0 is a rewrite of the original, widely-used Morfessor 1.0 software, with well documented command-line tools and library interface. It includes new features such as semi-supervised learning, online training, and integrated evaluation code.

1 Introduction

In the morphological segmentation task, the goal is to segment words into morphemes, the smallest meaning-carrying units. Morfessor is a family of methods for unsupervised morphological segmentation. The first version of Morfessor, called Morfessor Baseline, was developed by Creutz and Lagus (2002) its software implementation, Morfessor 1.0, released by Creutz and Lagus (2005b). A number of Morfessor variants have been developed later, including Morfessor Categories-MAP (Creutz and Lagus, 2005a) and Allomorfessor (Virpioja et al., 2010). Even though these algorithms improve Morfessor Baseline in some areas, the Baseline version has stayed popular as a generally applicable morphological analyzer (Spiegler et al., 2008; Monson et al., 2010).

Over the past years, Morfessor has been used for a wide range of languages and applications. The applications include large vocabulary continuous speech recognition (e.g. Hirsimäki et al., 2006), machine translation (e.g. Virpioja et al., 2007), and speech retrieval (e.g. Arisoy et al., 2009). Morfessor is well-suited for languages with concatenative morphology, and the tested languages include Finnish and Estonian (Hirsimäki et al., 2009), German (El-Desoky Mousa et al., 2010), and Turkish (Arisoy et al., 2009).

Morfessor 2.0 is a new implementation of the Morfessor Baseline algorithm.[1] It has been written in a modular manner and released as an open source project with a permissive license to encourage extensions. This paper includes a summary of the Morfessor 2.0 software and a description of the demonstrations that will be held. An extensive description of the features in Morfessor 2.0, including experiments, is available in the report by Virpioja et al. (2013).

2 Morfessor model and algorithms

Models of the Morfessor family are generative probabilistic models that predict compounds and their analyses (segmentations) given the model parameters. We provide a brief overview of the methodology; Virpioja et al. (2013) should be referred to for the complete formulas and description of the model and its training algorithms.

Unlike older Morfessor implementations, Morfessor 2.0 is agnostic in regard to the actual data being segmented. In addition to morphological segmentation, it can handle, for example, sentence chunking. To reflect this we use the following generic terms: The smallest unit that can be split will be an *atom* (letter). A *compound* (word) is a sequence of atoms. A *construction* (morph) is a sequence of atoms contained inside a compound.

2.1 Model and cost function

The cost function of Morfessor Baseline is derived using maximum a posteriori estimation. That is, the goal is to find the most likely parameters $\boldsymbol{\theta}$

[1]Morfessor 2.0 can be downloaded from the Morpho project website (`http://www.cis.hut.fi/projects/morpho/`) or GitHub repository (`https://github.com/aalto-speech/morfessor`).

given the observed training data $\boldsymbol{D}_W$:

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{D}_W \mid \boldsymbol{\theta}) \qquad (1)$$

Thus we are maximizing the product of the model prior $p(\boldsymbol{\theta})$ and the data likelihood $p(\boldsymbol{D}_W \mid \boldsymbol{\theta})$. As usual, the cost function to minimize is set as the minus logarithm of the product:

$$L(\boldsymbol{\theta}, \boldsymbol{D}_W) = -\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{D}_W \mid \boldsymbol{\theta}). \quad (2)$$

During training, the data likelihood is calculated using a hidden variable that contains the current chosen analyses. Secondly, it is assumed that the constructions in a compound occur independently. This simplifies the data likelihood to the product of all construction probabilities in the chosen analyses. Unlike previous versions, Morfessor 2.0 includes also the probabilities of the compound boundaries in the data likelihood.

For prior probability, Morfessor Baseline defines a distribution over the lexicon of the model. The prior assigns higher probability to lexicons that store fewer and shorter constructions. The lexicon prior consists of to parts, a product over the *form* probabilities and a product over the *usage* probabilities. The former includes the probability of a sequence of atoms and the latter the maximum likelihood estimates of the constructions. In contrast to Morfessor 1.0, Morfessor 2.0 currently supports only an implicit exponential length prior for the constructions.

2.2 Training and decoding algorithms

A Morfessor model can be trained in multiple ways. The standard batch training uses a local search utilizing recursive splitting. The model is initialized with the compounds and the full model cost is calculated. The data structures are designed in such way that the cost is efficient compute during the training.

In one epoch of the algorithm, all compounds in the training data are processed. For each compound, all possible two-part segmentations are tested. If one of the segmentations yields the lowest cost, it is selected and the segmentation is tried recursively on the resulting segments. In each step of the algorithm, the cost can only decrease or stay the same, thus guaranteeing convergence. The algorithm is stopped when the cost decreases less than a configurable threshold value in one epoch.

An extension of the Viterbi algorithm is used for decoding, that is, finding the optimal segmentations for new compound forms without changing the model parameters.

3 New features in Morfessor 2.0

3.1 Semi-supervised extensions

One important feature that has been implemented in Morfessor 2.0 are the semi-supervised extensions as introduced by Kohonen et al. (2010)

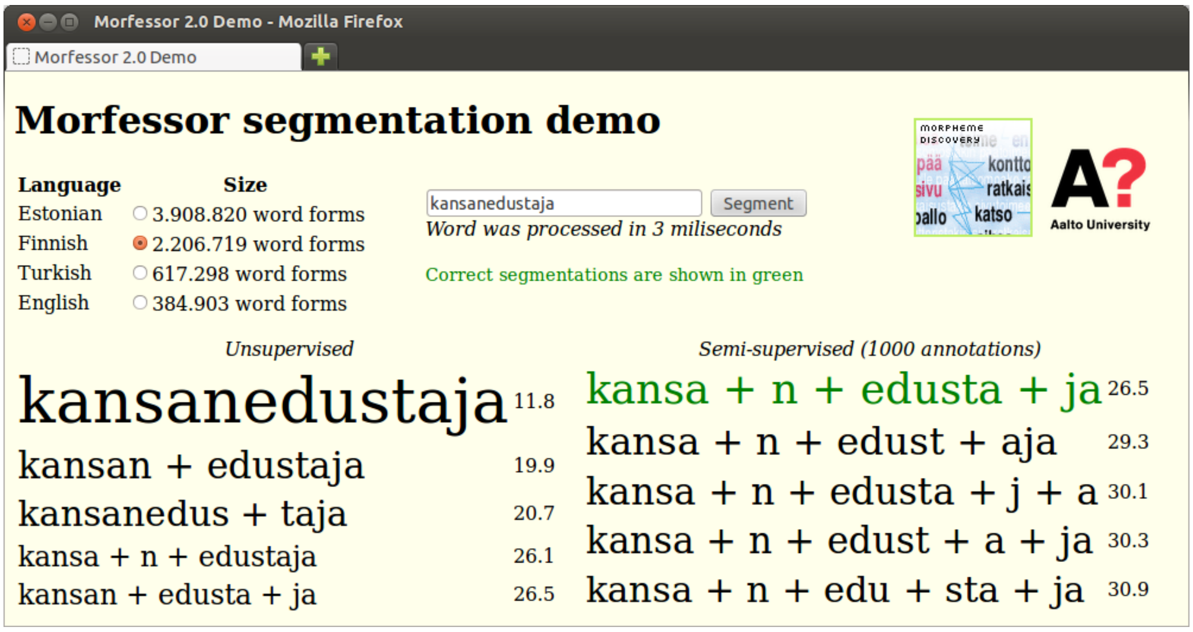
Morfessor Baseline tends to undersegment when the model is trained for morphological segmentation using a large corpus (Creutz and Lagus, 2005b). Oversegmentation or undersegmentation of the method are easy to control heuristically by including a weight parameter α for the likelihood in the cost function. A low α increases the priors influence, favoring small construction lexicons, while a high value increases the data likelihood influence, favoring longer constructions.

In semi-supervised Morfessor, the likelihood of an annotated data set is added to the cost function. As the amount of annotated data is typically much lower than the amount of unannotated data, its effect on the cost function may be very small compared to the likelihood of the unannotated data. To control the effect of the annotations, a separate weight parameter β can be included for the annotated data likelihood.

If separate development data set is available for automatic evaluation of the model, the likelihoods weights can be optimized to give the best output. This can be done by brute force using a grid search. However, Morfessor 2.0 implementation includes a simple heuristic for automatically tuning the value of α during the training, trying to balance precision and recall. A simple heuristic, which gives an equivalent contribution to the annotated data, is used for β.

3.2 On-line training

In addition to the batch training mode, Morfessor 2.0 supports on-line training mode, in which unannotated text is processed one compound at a time. This makes it simple to, for example, adapt pre-trained models for new type of data. As frequent compounds are encountered many times in running text, Morfessor 2.0 includes an option for randomly skipping compounds and constructions that have been recently analyzed. The random


Morfessor 2.0 Demo - Mozilla Firefox

Morfessor 2.0 Demo

Morfessor segmentation demo

Language	Size
Estonian	○ 3.908.820 word forms
Finnish	● 2.206.719 word forms
Turkish	○ 617.298 word forms
English	○ 384.903 word forms

kansanedustaja Segment

Word was processed in 3 miliseconds

Correct segmentations are shown in green

Unsupervised		*Semi-supervised (1000 annotations)*	
kansanedustaja	11.8	kansa + n + edusta + ja	26.5
kansan + edustaja	19.9	kansa + n + edust + aja	29.3
kansanedus + taja	20.7	kansa + n + edusta + j + a	30.1
kansa + n + edustaja	26.1	kansa + n + edust + a + ja	30.3
kansan + edusta + ja	26.5	kansa + n + edu + sta + ja	30.9

Figure 1: Screenshot from the Morfessor 2.0 demo.

skips can also be used to speed up the batch training.

3.3 Integrated evaluation code

One common method for evaluating the performance of a Morfessor model is to compare it against a gold standard segmentation using segmentation boundary precision and recall. To make the evaluation easy, the necessary tools for calculating the BPR metric by (Virpioja et al., 2011) are included in Morfessor 2.0. For significance testing when comparing multiple models, we have included the Wilcoxon signed-rank test. Both the evaluation code and statistical testing code are accessible from both the command line and the library interface.

3.4 N-best segmentation

In order to generate multiple segmentations for a single compound, Morfessor 2.0 includes a n-best Viterbi algorithm. It allows extraction of all possible segmentations for a compound and the probabilities of the segmentations.

4 Demonstration

In the demonstration session, multiple features and usages of Morfessor will be shown.

4.1 Web-based demonstration

A live demonstration will be given of segmenting text with Morfessor 2.0 for different language and training data options. In a web interface, the user can choose a language, select the size of the training corpus and other options. After that a word can be given which will be segmented using n-best Viterbi, showing the 5 best results.

A list of planned languages can be found in Table 1. A screen shot of the demo interface is shown in Figure 1.

Languages	# Words	# Word forms
English	62M	384.903
Estonian	212M	3.908.820
Finnish	36M	2.206.719
German	46M	1.266.159
Swedish	1M	92237
Turkish	12M	617.298

Table 1: List of available languages for Morfessor 2.0 demonstration.

4.2 Command line interface

The new command line interface will be demonstrated to train and evaluate Morfessor models from texts in different languages. A diagram of the tools is shown in Figure 2

4.3 Library interface

Interfacing with the Morfessor 2.0 Python library will be demonstrated for building own scientific experiments, as well as integrating Morfessor in

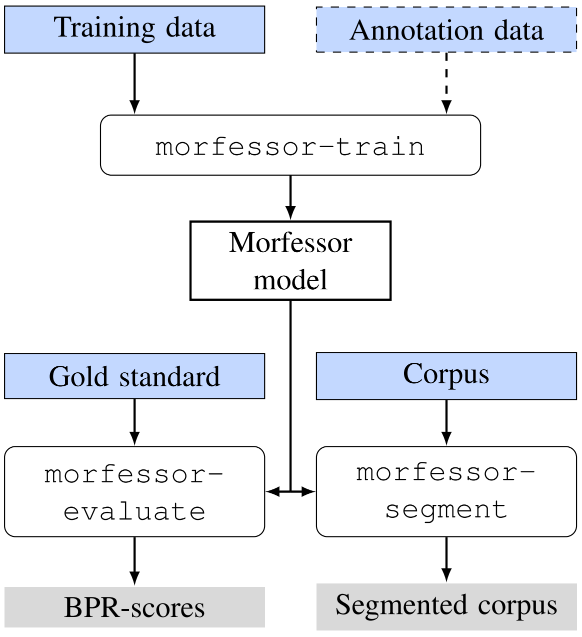


Figure 2: The standard workflow for Morfessor command line tools

bigger project. Also the code of the Web based demonstration will be shown as an example.

Acknowledgements

The authors have received funding from the EC's 7th Framework Programme (FP7/2007–2013) under grant agreement n°287678 and the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170) and the LASTU Programme (grants n°256887 and 259934). The experiments were performed using computer resources within the Aalto University School of Science "Science-IT" project.

References

E. Arisoy, D. Can, S. Parlak, H. Sak, and M. Saraclar. 2009. Turkish broadcast news transcription and retrieval. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(5):874–883.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In Mike Maxwell, editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics, July.

M. Creutz and K. Lagus. 2005a. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland, June. Helsinki University of Technology.

M. Creutz and K. Lagus. 2005b. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

A. El-Desoky Mousa, M. Ali Basha Shaik, R. Schluter, and H. Ney. 2010. Sub-lexical language models for German LVCSR. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 171–176. IEEE.

T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech & Language*, 20(4):515–541.

T. Hirsimäki, J. Pylkkönen, and M. Kurimo. 2009. Importance of high-order n-gram models in morph-based speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):724–732.

O. Kohonen, S. Virpioja, and K. Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86, Uppsala, Sweden, July. Association for Computational Linguistics.

C. Monson, K. Hollingshead, and B. Roark. 2010. Simulating morphological analyzers with stochastic taggers for confidence estimation. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 649–657. Springer.

S. Spiegler, B. Golénia, K. Shalonova, P. Flach, and R. Tucker. 2008. Learning the morphology of zulu with different degrees of supervision. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 9–12. IEEE.

S. Virpioja, J. Väyrynen, M. Creutz, and M. Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *Proceedings of the Machine Translation Summit XI*, pages 491–498, Copenhagen, Denmark, September.

S. Virpioja, O. Kohonen, and K. Lagus. 2010. Unsupervised morpheme analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of *LNCS*, pages 609–616. Springer Berlin / Heidelberg.

S. Virpioja, V. Turunen, S. Spiegler, O. Kohonen, and M. Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *TAL*, 52(2):45–90.

S. Virpioja, P. Smit, S. Grönroos, and M. Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Aalto University, Finland.

CASMACAT: A Computer-assisted Translation Workbench

V. Alabau[⋆], **C. Buck**[‡], **M. Carl**[†], **F. Casacuberta**[⋆], **M. García-Martínez**[†]
U. Germann[‡], **J. González-Rubio**[⋆], **R. Hill**[‡], **P. Koehn**[‡], **L. A. Leiva**[⋆]
B. Mesa-Lao[†], **D. Ortiz**[⋆], **H. Saint-Amand**[‡], **G. Sanchis**[⋆], **C. Tsoukala**[‡]

[⋆]PRHLT Research Center, Universitat Politècnica de València
{valabau,fcn,jegonzalez,luileito,dortiz,gsanchis}@dsic.upv.es

[†]Copenhagen Business School, Department of International Business Communication
{ragnar.bonk,mc.isv,mgarcia,bm.ibc}@cbs.dk

[‡]School of Informatics, University of Edinburgh
{cbuck,ugermann,rhill2,pkoehn,hsamand,ctsoukal}@inf.ed.ac.uk

Abstract

CASMACAT is a modular, web-based translation workbench that offers advanced functionalities for computer-aided translation and the scientific study of human translation: automatic interaction with machine translation (MT) engines and translation memories (TM) to obtain raw translations or close TM matches for conventional post-editing; interactive translation prediction based on an MT engine's search graph, detailed recording and replay of edit actions and translator's gaze (the latter via eye-tracking), and the support of e-pen as an alternative input device. The system is open source sofware and interfaces with multiple MT systems.

1 Introduction

CASMACAT[1] (Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation) is a three-year project to develop an advanced, interactive workbench for computer-assisted translation (CAT). Currently, at the end of the second year, the tool includes an array of innovative features that combine to offer a rich, user-focused working environment not available in any other CAT tool.

CASMACAT works in close collaboration with the MATECAT project[2], another open-source web-based CAT tool. However, while MATECAT is concerned with conventional CAT, CASMACAT is focused on enhancing user interaction and facilitating the real-time involvement of human translators. In particular, CASMACAT provides highly interactive editing and logging features.

[1]http://www.casmacat.eu

[2]http://www.matecat.com

Through this combined effort, we hope to foster further research in the area of CAT tools that improve the translation workflow while appealing to both professional and amateur translators without advanced technical skills.

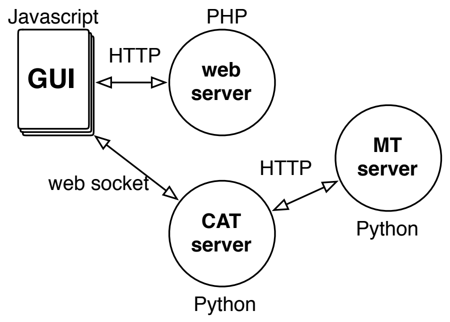
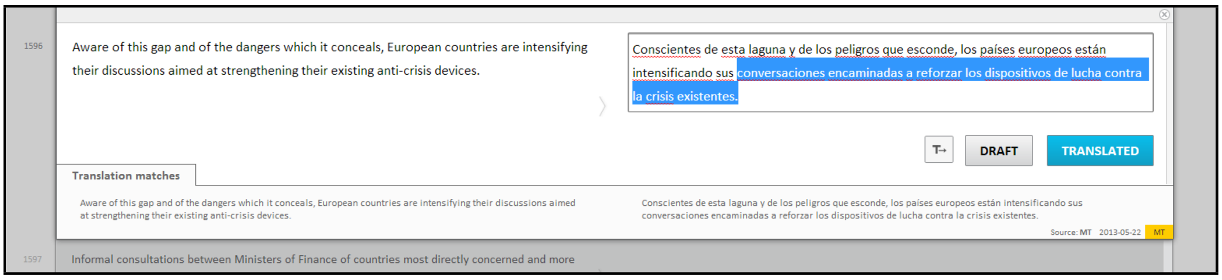


Figure 1: Modular design of the workbench: Web-based components (GUI and web server), CAT server and MT server can be swapped out.

2 Design and components

The overall design of the CASMACAT workbench is modular. The system consists of four components. (1) a front-end GUI implemented in HTML5 and JavaScript; (2) a back-end implemented in PHP; (3) a CAT server that manages the editing process and communicates with the GUI through web sockets; (4) a machine translation (MT) server that provides raw translation of source text as well as additional information, such as a search graph that efficiently encodes alternative translation options. Figure 1 illustrates how these components interact with each other. The CAT and MT servers are written in Python and interact with a number of software components implemented in C++. All recorded information (source, translations, edit logs) is permanently stored in a MySQL database.

These components communicate through a well-defined API, so that alternative implementations can be used. This modular architecture al-


1596 Aware of this gap and of the dangers which it conceals, European countries are intensifying their discussions aimed at strengthening their existing anti-crisis devices.

Conscientes de esta laguna y de los peligros que esconde, los países europeos están intensificando sus conversaciones encaminadas a reforzar los dispositivos de lucha contra la crisis existentes.

Aware of this gap and of the dangers which it conceals, European countries are intensifying their discussions aimed at strengthening their existing anti-crisis devices.

Conscientes de esta laguna y de los peligros que esconde, los países europeos están intensificando sus conversaciones encaminadas a reforzar los dispositivos de lucha contra la crisis existentes.

Source: MT 2013-05-22 MT

1597 Informal consultations between Ministers of Finance of countries most directly concerned and more

Figure 2: Translation view for an interactive post-editing task.

lows the system to be used partially. For instance, the CAT and MT servers can be used separately as part of a larger translation workflow, or only as a front-end when an existing MT solution is already in place.

2.1 CAT server

Some of the interactive features of CASMACAT require real-time interaction, such as interactive text-prediction (ITP), so establishing an HTTP connection every time would cause a significant network overhead. Instead, the CAT server relies on web sockets, by means of Python's Tornadio.

When interactive translation prediction is enabled, the CAT server first requests a translation together with the search graph of the current segment from the MT server. It keeps a copy of the search graph and constantly updates and visualizes the translation prediction based on the edit actions of the human translator.

2.2 MT server

Many of the functions of the CAT server require information from an MT server. This information includes not only the translation of the input sentence, but also n-best lists, search graphs, word alignments, and so on. Currently, the CASMACAT workbench supports two different MT servers: Moses (Koehn et al., 2007) and Thot (Ortiz-Martínez et al., 2005).

The main call to the MT server is a request for a translation. The request includes the source sentence, source and target language, and optionally a user ID. The MT server returns an JSON object, following an API based on *Google Translate*.

3 Graphical User Interface

Different *views*, based on the MATECAT GUI, perform different tasks. The translation view is the primary one, used when translating or post-editing, including logging functions about the translation/post-editing process. Other views implement interfaces to upload new documents or to manage the documents that are already in the system. Additionally, a replay view can visualize all edit actions for a particular user session, including eye tracking information, if available.

3.1 Post-Editing

In the translation view (Figure 2), the document is presented in segments and the assistance features provided by CASMACAT work at the segment level. If working in a post-editing task without ITP, up to three MT or TM suggestions are provided for the user to choose. Keyboard shortcuts are available for performing routine tasks, for instance, loading the next segment or copying source text into the edit box. The user can assign different status to each segment, for instance, *"translated"* for finished ones or *"draft"* for segments that still need to be reviewed. Once finished, the translated document can be downloaded in XLIFF format.[3]

In the translation view, all user actions related to the translation task (e.g. typing activity, mouse moves, selection of TM proposals, etc.) are recorded by the logging module, collecting valuable information for off-line analyses.

3.2 Interactive Translation Prediction

Here we briefly describe the main advanced CAT features implemented in the workbench so far.

Intelligent Autocompletion: ITP takes place every time a keystroke is detected by the system (Barrachina et al., 2009). In such event, the system produces a prediction for the rest of the sentence according to the text that the user has already entered. This prediction is placed at the right of the text cursor.

Confidence Measures: Confidence measures (CMs) have two main applications in

[3]XLIFF is a popular format in the translation industry.

MT (González-Rubio et al., 2010). Firstly, CMs allow the user to clearly spot wrong translations (e.g., by rendering in red those translations with very low confidence according to the MT module). Secondly, CMs can also inform the user about the translated words that are dubious, but still have a chance of being correct (e.g., rendered in orange). Figure 3 illustrates this.

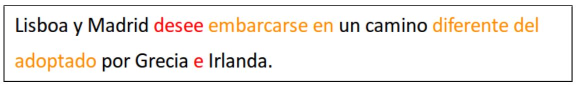

Lisboa y Madrid desee embarcarse en un camino diferente del adoptado por Grecia e Irlanda.

Figure 3: Visualisation of Confidence Measures

Prediction Length Control: Providing the user with a new prediction whenever a key is pressed has been proved to be cognitively demanding (Alabau et al., 2012). Therefore, the GUI just displays the prediction up to the first wrong word according to the CMs provided by the system (Figure 4).

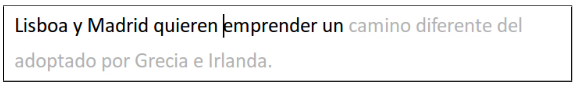
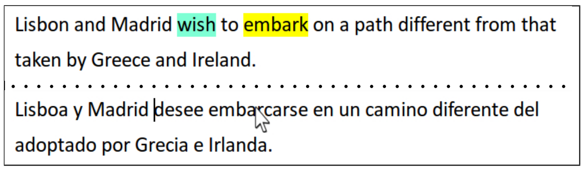

Lisboa y Madrid quieren emprender un camino diferente del adoptado por Grecia e Irlanda.

Figure 4: Prediction Length Control

Search and Replace: Most of CAT tools provide the user with intelligent search and replace functions for fast text revision. CASMACAT features a straightforward function to run search and replacement rules on the fly.

Word Alignment Information: Alignment of source and target words is an important part of the translation process (Brown et al., 1993). To display their correspondence, they are hihglighted every time the user places the mouse or the text cursor on a word; see Figure 5.


Lisbon and Madrid wish to embark on a path different from that taken by Greece and Ireland.

Lisboa y Madrid desee embarcarse en un camino diferente del adoptado por Grecia e Irlanda.

Figure 5: Visualisation of Word Alignment

Prediction Rejection: With the purpose of easing user interaction, CASMACAT also supports a one-click rejection feature (Sanchis-Trilles et al., 2008). This feature invalidates the current prediction made for the sentence that is being translated, and provides the user with an alternate one.

3.3 Replay mode and logging functions

The CASMACAT workbench implements detailed logging of user activity data, which enables both automatic analysis of translator behaviour and retrospective replay of a user session. Replay takes place in the translation view of the GUI and it displays the screen status of the recorded translation/post-editing process. The workbench also features a plugin to enrich the replay mode with gaze data coming from an eye-tracker. This eye-tracking integration is possible through a project-developed web browser extension which, at the moment, has only been fully tested with SR-Research EyeLinks[4].

4 E-pen Interaction

E-pen interaction is intended to be a complementary input rather than a substitution of the keyboard. The GUI features the minimum components necessary for e-pen interaction; see Figure 6. When the e-pen is enabled, the display of the current segment is changed so that the source segment is shown above the target segment. Then the drawing area is maximised horizontally, facilitating handwriting, particularly in tablet devices. An HTML canvas is also added over the target segment, where the user's drawings are handled. This is achieved by means of MINGESTURES (Leiva et al., 2013), a highly accurate, high-performance gesture set for interactive text editing that can distinguish between gestures and handwriting. Gestures are recognised on the client side so the response is almost immediate. Conversely, when handwritten text is detected, the pen strokes are sent to the server. The hand-written text recognition (HTR) server is based on iAtros, an open source HMM decoder.

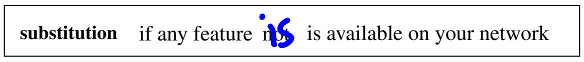


Figure 6: Word substitution with e-pen interaction

5 Evaluation

The CASMACAT workbench was recently evaluated in a field trial at Celer Soluciones SL, a language service provider based in Spain. The trial involved nine professional translators working with the workbench to complete different post-editing tasks from English into Spanish. The pur-

[4] http://www.sr-research.com

pose of this evaluation was to establish which of the workbench features are most useful to professional translators. Three different configurations were tested:

- `PE`: The CASMACAT workbench was used only for conventional post-editing, without any additional features.
- `IA`: Only the Intelligent Autocompletion feature was enabled. This feature was tested separately because it was observed that human translators substantially change the way they interact with the system.
- `ITP`: All features described in Section 3.2 were included in this configuration, excepting CMs, which were deemed to be not accurate enough for use in a human evaluation.

For each configuration, we measured the average time taken by the translator to produce the final translation (on a segment basis), and the average number of edits required to produce the final translation. The results are shown in Table 1.

Setup	**Avg. time (s)**	**Avg. # edits**
`PE`	92.2 ± 4.82	141.39 ± 7.66
`IA`	86.07 ± 4.92	124.29 ± 7.28
`ITP`	123.3 ± 29.72	137.22 ± 13.67

Table 1: Evaluation of the different configurations of the CASMACAT workbench. Edits are measured in keystrokes, i.e., insertions and deletions.

While differences between these numbers are not statistically significant, the apparent slowdown in translation with ITP is due to the fact that all translators had experience in post-editing but none of them had ever used a workbench featuring intelligent autocompletion before. Therefore, these were somewhat unsurprising results.

In a post-trial survey, translators indicated that, on average, they liked the `ITP` system the best. They were not fully satisfied with the freedom of interactivity provided by the `IA` system. The lack of any visual aid to control the intelligent autocompletions provided by the system made translators think that they had to double-check any of the proposals made by the system when making only a few edits.

6 Conclusions

We have introduced the current CASMACAT workbench, a next-generation tool for computer assisted translation. Each of the features available in the most recent prototype of the workbench has been explained. Additionally, we have presented an executive report of a field trial that evaluated genuine users' performance while using the workbench. Although E-pen interaction has not yet been evaluated outside of the laboratory, it will the subject of future field trials, and a working demonstration is available.

Acknowledgements

Work supported by the European Union 7th Framework Program (FP7/2007-2013) under the CASMACAT project (grant agreement n^{o} 287576).

References

Vicent Alabau, Luis A. Leiva, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2012. User evaluation of interactive machine translation systems. In *Proc. EAMT*, pages 20–23.

Sergio Barrachina et al. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Peter Brown et al. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2010. On the use of confidence measures within an interactive-predictive machine translation system. In *Proc. of EAMT*.

Philipp Koehn et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*, pages 177–180.

Luis A. Leiva, Vicent Alabau, and Enrique Vidal. 2013. Error-proof, high-performance, and context-aware gestures for interactive text edition. In *Proc. of CHI*, pages 1227–1232.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. In *Proc. of MT Summit X*, pages 141–148.

G. Sanchis-Trilles et al. 2008. Improving interactive machine translation via mouse actions. In *Proc. of EMNLP*, pages 485–494.

Jane: Open Source Machine Translation System Combination

Markus Freitag[1] and **Matthias Huck**[2] and **Hermann Ney**[1]

[1] Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{freitag,ney}@cs.rwth-aachen.de

[2] School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB, UK
mhuck@inf.ed.ac.uk

Abstract

Different machine translation engines can be remarkably dissimilar not only with respect to their technical paradigm, but also with respect to the translation output they yield. *System combination* is a method for combining the output of multiple machine translation engines in order to take benefit of the strengths of each of the individual engines.

In this work we introduce a novel system combination implementation which is integrated into `Jane`, RWTH's open source statistical machine translation toolkit. On the most recent *Workshop on Statistical Machine Translation* system combination shared task, we achieve improvements of up to 0.7 points in BLEU over the best system combination hypotheses which were submitted for the official evaluation. Moreover, we enhance our system combination pipeline with additional *n*-gram language models and lexical translation models.

1 Introduction

We present a novel machine translation system combination framework which has been implemented and released as part of the most recent version of the `Jane` toolkit.[1] Our system combination framework has already been applied successfully for joining the outputs of different individual machine translation engines from several project partners within large-scale projects like Quaero (Peitz and others, 2013), EU-BRIDGE (Freitag and others, 2013), and DARPA BOLT. The combined translation is typically of better quality than any of the individual hypotheses. The source code of our framework has now been released to the public.

We focus on system combination via confusion network decoding. This basically means that we align all input hypotheses from individual machine translation (MT) engines together and extract a combination as a new output. For our baseline algorithm we only need the first best translation from each of the different MT engines, without any additional information. Supplementary to the baseline models integrated into our framework, we optionally allow for utilization of *n*-gram language models and IBM-1 lexicon models (Brown et al., 1993), both trained on additional training corpora that might be at hand.

We evaluate the `Jane` system combination framework on the latest official *Workshop on Statistical Machine Translation* (WMT) system combination shared task (Callison-Burch et al., 2011). Many state-of-the-art MT system combination toolkits have been evaluated on this task, which allows us to directly compare the results obtained with our novel `Jane` system combination framework with the best known results obtained with other toolkits.

The paper is structured as follows: We commence with giving a brief outline of some related work (Section 2). In Section 3 we describe the techniques which are implemented in the `Jane` MT system combination framework. The experimental results are presented and analyzed in Section 4. We conclude the paper in Section 5.

2 Related Work

The first application of system combination to MT has been presented by Bangalore et al. (2001). They used a multiple string alignment (MSA) approach to align the hypotheses together and built a confusion network from which the system combination output is determined using majority vot-

[1] `Jane` is publicly available under an open source non-commercial license and can be downloaded from `http://www.hltpr.rwth-aachen.de/jane/`.

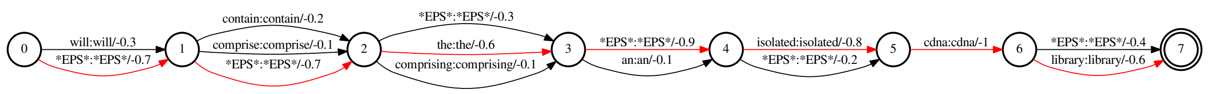


Figure 1: Scored confusion network. *EPS* denotes the empty word, red arcs highlight the shortest path.

ing and an additional language model. Matusov et al. (2006) proposed an alignment based on the GIZA++ toolkit which introduced word reordering not present in MSA, and Sim et al. (2007) used alignments produced by TER scoring (Snover et al., 2006). Extensions of the last two are based on hidden Markov models (He et al., 2008), inversion transduction grammars (Karakos et al., 2008), or METEOR (Heafield and Lavie, 2010).

3 The `Jane` MT System Combination Framework

In this section we describe the techniques for MT system combination which we implemented in the `Jane` toolkit.[2] We first address the generation of a confusion network from the input translations. For that we need a pairwise alignment between all input hypotheses. We then present word reordering mechanisms, the baseline models, and additional advanced models which can be applied for system combination using `Jane`. The system combination decoding step basically involves determining the shortest path through the confusion network based on several model scores from this network.

3.1 Confusion Network

A confusion network represents all different combined translations we can generate from the set of provided input hypotheses. Figure 1 depicts an example of a confusion network. A word alignment between all pairs of input hypotheses is required for generating a confusion network. For convenience, we first select one of the input hypotheses as the primary hypothesis. The primary hypothesis then determines the word order and all remaining hypotheses are word-to-word aligned to the given word order.

To generate a meaningful confusion network, we should adopt an alignment which only allows to switch between words which are synonyms, misspellings, morphological variants or on a higher level paraphrases of the words from the primary hypothesis. In this work we use METEOR alignments. METEOR (Denkowski and Lavie, 2011) was originally designed to reorder a translation for scoring and has a high precision. The recall is lower because synonyms which are not in the METEOR database or punctuation marks like "!" and "?" are not aligned to each other. For our purposes, we augment the METEOR paraphrase table with entries like ".|!", ".|?", or "the|a".

Figure 2 shows an example METEOR hypothesis alignment. The primary hypothesis "isolated cdna lib" determines the word order. An entry "a|b" means that word "a" from a secondary hypothesis has been aligned to word "b" from the primary one. "*EPS*" is the empty word and thus an entry "*EPS*|b" means that no word could be aligned to the primary hypothesis word "b". "a|*EPS*" means that the word "a" has not been aligned to any word from the primary hypothesis.

After producing the alignment information, we can build the confusion network. Now, we are able to not only extract the original primary hypothesis from the confusion network but also switch words from the primary hypothesis to words from any secondary hypothesis (also the empty word) or insert words or sequences of words.

In the final confusion network, we do not stick to one hypothesis as the primary system. For m input hypotheses we build m different confusion networks, each having a different system as primary system. The final confusion network is a union of all m networks.[3]

The most straightforward way to obtain a combined hypothesis from a confusion network is to extract it via majority voting. For example, in the first column in Figure 3, "the" has been seen three times, but the translation options "a" and "an" have each been seen only once. By means of a straight majority vote we would extract "the". As the different single system translations are of varying utility for system combination, we assign a system weight to each input hypothesis. The system weights are set by optimizing scaling factors for binary system voting features (cf. Section 3.3). We employ some more weighted baseline features

[2]Practical usage aspects are explained in the manual: `http://www.hltpr.rwth-aachen.de/jane/manual.pdf`

[3]`Jane`'s implementation for building confusion networks is based on the OpenFST library (Allauzen et al., 2007).

the|*EPS*	isolated|isolated	cdna|cdna	*EPS*|lib
a|*EPS*	isolated|isolated	cdna|cdna	lib|lib
an|*EPS*	isolated|isolated	cdna|cdna	lib|lib
the|*EPS*	*EPS*|isolated	cdna|cdna	*EPS*|lib
the|*EPS*	*EPS*|isolated	cdna|cdna	lib|lib

Figure 2: Alignment result after running METEOR. *EPS* denotes the empty word.

EPS	isolated	cdna	lib
the	isolated	cdna	*EPS*
a	isolated	cdna	lib
an	isolated	cdna	lib
the	*EPS*	cdna	lib
the	*EPS*	cdna	*EPS*
the	isolated	cdna	lib

Figure 3: Majority vote on aligned words. The last line is the system combination output.

and additional models (cf. Section 3.4) in the decision process. In Figure 1 we scored the confusion network with some system weights and used the shortest path algorithm to find the hypothesis with the highest score (the hypothesis along the path highlighted in red).

3.2 Word Reordering

Many words from secondary hypotheses can be unaligned as they have no connection to any words of the primary hypothesis. However, words from different secondary systems could be related to each other. In order to account for these relations and to give the words from the secondary hypotheses a higher chance to be present in the combined output, we introduce some simple word reordering mechanisms.

We rank the hypotheses according to a language model trained on all input hypotheses. We initialize the confusion network with the sentence from the primary system. During the generation of the confusion network we align the hypotheses consecutively into the confusion network via the following procedure:

- If a word w_i from hypothesis A has a relation to a word v_j of the primary hypothesis, we insert it as a new translation alternative to v_j.
- If w_i has no relation to the primary, but to a word u_k from a secondary hypothesis in the confusion network, we insert w_i as a new translation alternative to u_k.
- Otherwise we insert w_i in front of the previous inserted word w_{i-1} of hypothesis A. The new position gets an epsilon arc for the primary and all unrelated secondary systems.

3.3 Baseline Models

Once we have the final confusion network, we want to adopt models which are valuable features to score the different translation options. In our implementation we use the following set of standard models:

***m* binary system voting features** For each word the voting feature for system i $(1 \leq i \leq m)$ is 1 iff the word is from system i, otherwise 0.

Binary primary system feature A feature that marks the primary hypothesis.

LM feature 3-gram language model trained on the input hypotheses.

Word penalty Counts the number of words.

3.4 Additional Models

The `Jane` system combination toolkit also provides the possibility to utilize some additional models for system combination. For the current release we integrated the optional usage of the following additional models:

Big LM A big language model trained on larger monolingual target-side corpora.

IBM-1 Source-to-target and target-to-source IBM-1 lexical translation models obtained from bilingual training data.

4 Experimental Results

All experiments are conducted on the latest official WMT system combination shared task.[4] We exclusively employ resources which were permitted for the constrained track of the task in all our setups. The big LM was trained on News Commentary and Europarl data. As tuning set we used *newssyscombtune2011*, as test set we used *newssyscombtest2011*. Feature weights have been optimized with MERT (Och, 2003). Table 1 contains the empirical results (truecase). For all four language pairs we achieve improvements over the best 2011 evaluation system combination submission either in BLEU or TER. We get the highest improvement of 0.7 points in BLEU for es→en when adding both the big LM and IBM-1 features. Adding the big LM over the baseline enhances the translation quality for all four language pairs. Adding IBM-1 lexicon models on top of the big LM is of marginal or no benefit for most language

[4]The most recent system combination shared task that has been organized as part of the WMT evaluation campaign took place in 2011. `http://www.statmt.org/wmt11/system-combination-task.html`

Table 1: Experimental results on the WMT system combination tasks (newssyscombtest2011).

system	**cz→en**		**de→en**		**es→en**		**fr→en**	
	Bleu	Ter	Bleu	Ter	Bleu	Ter	Bleu	Ter
best single system	28.7	53.4	23.0	59.5	28.9	51.2	29.4	52.0
best 2011 evaluation syscomb	28.8	55.2	25.1	57.4	32.4	49.9	31.3	50.1
`Jane` syscomb baseline	28.8	53.6	24.7	57.6	32.7	50.3	31.3	50.3
`Jane` syscomb + big LM	29.0	54.5	25.0	57.3	32.9	50.3	31.4	50.0
`Jane` syscomb + big LM + IBM-1	29.0	54.5	25.0	57.3	33.1	50.0	31.5	50.1

pairs, but at least provides slight improvements for es→en.

5 Conclusion

RWTH's open source machine translation toolkit `Jane` now includes a state-of-the-art system combination framework. We found that the `Jane` system combination performs on a similar level or better than the best evaluation system combination submissions on all WMT 2011 system combination shared task language pairs (with English as target language). We furthermore presented the effects of integrating a big *n*-gram language model and of lexical features from IBM-1 models.

Acknowledgements

This material is based upon work supported by the DARPA BOLT project under Contract No. HR0011- 12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 287658.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In Jan Holub and Jan Zdárek, editors, *Implementation and Application of Automata*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer Berlin Heidelberg.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing Consensus Translation from Multiple Machine Translation Systems. In *Proc. of ASRU*, pages 351–354.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F. Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proc. of WMT*, pages 22–64.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proc. of WMT*, pages 85–91.

Markus Freitag et al. 2013. EU-BRIDGE MT: Text Translation of Talks in the EU-BRIDGE Project. In *Proc. of IWSLT*.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems. In *Proc. of EMNLP*, pages 98–107.

Kenneth Heafield and Alon Lavie. 2010. Combining Machine Translation Output with Open Source: The Carnegie Mellon Multi-Engine Machine Translation Scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine translation system combination using ITG-based alignments. In *Proc. of ACL: Short Papers*, pages 81–84.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Proc. of EACL*, pages 33–40.

Franz J. Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of ACL*, pages 160–167.

Stephan Peitz et al. 2013. Joint WMT 2013 Submission of the QUAERO Project. In *Proc. of WMT*, pages 185–192.

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus Network Decoding for Statistical Machine Translation System Combination. In *Proc. of ICASSP*, volume 4, pages 105–108.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proc. of AMTA*, pages 223–231.

CHISPA on the GO
A mobile Chinese-Spanish translation service for travelers in trouble

Jordi Centelles[1,2], **Marta R. Costa-jussà**[1,2] **and Rafael E. Banchs**[2]
[1] Universitat Politècnica de Catalunya, Barcelona
[2] Institute for Infocomm Research, Singapore
{visjcs,vismrc,rembanchs}@i2r.a-star.edu.sg

Abstract

This demo showcases a translation service that allows travelers to have an easy and convenient access to Chinese-Spanish translations via a mobile app. The system integrates a phrase-based translation system with other open source components such as Optical Character Recognition and Automatic Speech Recognition to provide a very friendly user experience.

1 Introduction

During the last twenty years, Machine Translation technologies have matured enough to get out from the academic world and jump into the commercial area. Current commercially available machine translation services, although still not good enough to replace human translations, are able to provide useful and reliable support in certain applications such as cross-language information retrieval, cross-language web browsing and document exploration.

On the other hand, the increasing use of smartphones, their portability and the availability of internet almost everywhere, have allowed for lots of traditional on-line applications and services to be deployed on these mobile platforms.

In this demo paper we describe "CHISPA on the GO" a Chinese-Spanish translation service that intends to provide a portable and easy to use language assistance tool for travelers between Chinese and Spanish speaking countries.

The main three characteristics of the presented demo system are as follows:

- First, the system uses a direct translation between Chinese and Spanish, rather than using a pivot language as intermediate step as most of the current commercial systems do when dealing with distant languages.
- Second, in addition to support on-line translations, as other commercial systems, our system also supports access from mobile platforms, Android and iOS, by means of native mobile apps.
- Third, the mobile apps combine the base translation technology with other supporting technologies such as Automatic Speech Recognition (ASR), Optical Character Recognition (OCR), Image retrieval and Language detection in order to provide a friendly user experience.

2 SMT system description

The translation technology used in our system is based on the well-known phrase-based translation statistical approach (Koehn et al., 2003). This approach performs the translation splitting the source sentence in segments and assigning to each segment a bilingual phrase from a phrase-table. Bilingual phrases are translation units that contain source words and target words, and have different scores associated to them. These bilingual phrases are then selected in order to maximize a linear combination of feature functions. Such strategy is known as the log-linear model (Och and Ney, 2002). The two main feature functions are the translation model and the target language model. Additional models include lexical weights, phrase and word penalty and reordering.

2.1 Experimental details

Generally, Chinese-Spanish translation follows pivot approaches to be translated (Costa-jussà et al., 2012) because of the lack of parallel data to train the direct approach. The main advantage of our system is that we are using the direct approach and at the same time we rely on a pretty large corpus. For Chinese-Spanish, we use (1) the *Holy Bible* corpus (Banchs and Li, 2008), (2) the

United Nations corpus, which was released for research purposes (Rafalovitch and Dale, 2009), (3) a small subset of the European Parliament Plenary Speeches where the Chinese part was synthetically produced by translating from English, (4) a large TAUS corpus (TausData, 2013) which comes from technical translation memories, and (5) an in-house developed small corpus in the transportation and hospitality domains. In total we have 70 million words.

A careful preprocessing was developed for all languages. Chinese was segmented with Stanford segmenter (Tseng et al., 2005) and Spanish was preprocessed with Freeling (Padró et al., 2010). When Spanish is used as a source language, it is preprocessed by lower-casing and unaccented the input. Finally, we use the MOSES decoder (Koehn et al., 2007) with standard configuration: align-grow-final-and alignment symmetrization, 5-gram language model with interpolation and kneser-ney discount and phrase-smoothing and lexicalized reordering. We use our in-house developed corpus to optimize because our application is targeted to the travelers-in-need domain.

3 Web Translator and Mobile Application

This section describes the main system architecture and the main features of web translator and the mobile applications.

3.1 System architecture

Figure 1 shows a block diagram of the system architecture. Below, we explain the main components of the architecture, starting with the back-end and ending with the front-end.

3.1.1 Back-end

As previously mentioned, our translation system uses MOSES. More specifically, we use the open source MOSES server application developed by Saint-Amand (2013). Because translation tables need to be kept permanently in memory, we use binary tables to reduce the memory space consumption. The MOSES server communicates with a PHP script that is responsible for receiving the query to be translated and sending the translation back.

For the Chinese-Spanish language pair, we count with four types of PHP scripts. Two of them communicate with the web-site and the other two with the mobile applications. In both cases, one

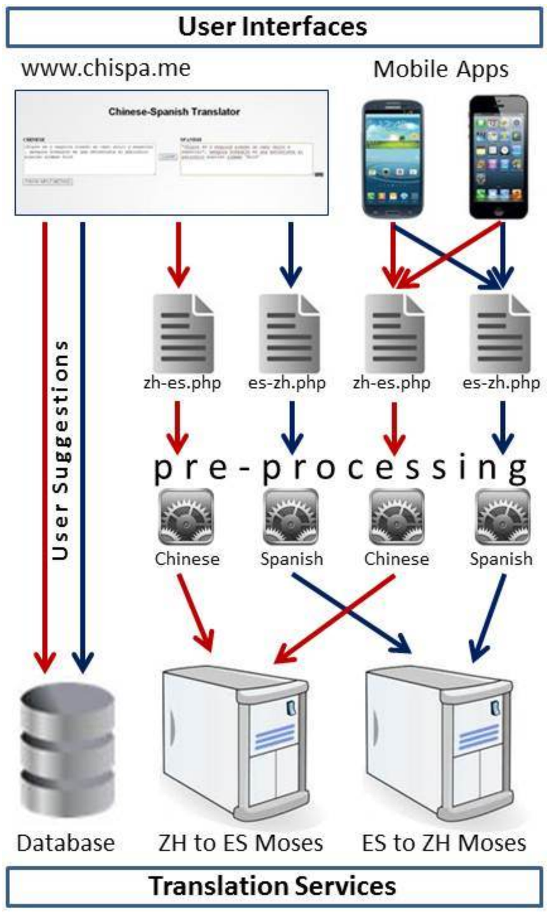


Figure 1: Block diagram of the system architecture

of the two PHP scripts supports Chinese to Spanish translations and the other one the Spanish to Chinese translations.

The functions of the PHP scripts responsible for supporting translations are: (1) receive the Chinese/Spanish queries from the front-end; (2) preprocess the Chinese/Spanish queries; (3) send these preprocessed queries to the Chinese/Spanish to Spanish/Chinese MOSES servers; (4) receive the translated queries; and (5) send them back to the front-end.

3.1.2 Front-end

HTML and Javascript constitute the main code components of the translation website.Another web development technique used was Ajax, which allows for asynchronous communication between the MOSES server and the website. This means that the website does not need to be refreshed after every translation.

The HTTP protocol is used for the communications between the web and the server. Specifically, we use the POST method, in which the server receives data through the request message's body.

The Javascript is used mainly to implement the input methods of the website, which are a Spanish keyboard and a Pinyin input method, both open source and embedded into our code. Also, using Javascript, a small delay was programmed in order to automatically send the query to the translator each time the user stops typing.

Another feature that is worth mentioning is the support of user feedback to suggest better translations. Using MYSQL, we created a database in the server where all user suggestions are stored. Later, these suggestions can be processed off-line and used in order to improve the system.

Additionally, all translations processed by the system are stored in a file. This information is to be exploited in the near future, when a large number of translations has been collected, to mine for the most commonly requested translations. The most common translation set will be used to implement an index and search engine so that any query entered by a user, will be first checked against the index to avoid overloading the translation engine.

3.2 Android and iphone applications

The android app was programmed with the Android development tools (ADT). It is a plug-in for the Eclipse IDE that provides the necessary environment for building an app.

The Android-based "CHISPA on the GO" app is depicted in Figure 2.

For the communication between the Android app and the server we use the HTTPClient interface. Among other things, it allows a client to send data to the server via, for instance, the POST method, as used on the website case.

For the Iphone app we use the xcode software provided by apple and the programming language used is Objective C.

In addition to the base translation system, the app also incorporates Automatic Speech Recognition (ASR), Optical Character Recognition technologies as input methods (OCR), Image retrieval and Language detection.

3.2.1 ASR and OCR

In the case of ASR, we relay on the native ASR engines of the used mobile platforms: Jelly-bean in the case of Android[1] and Siri in the case of

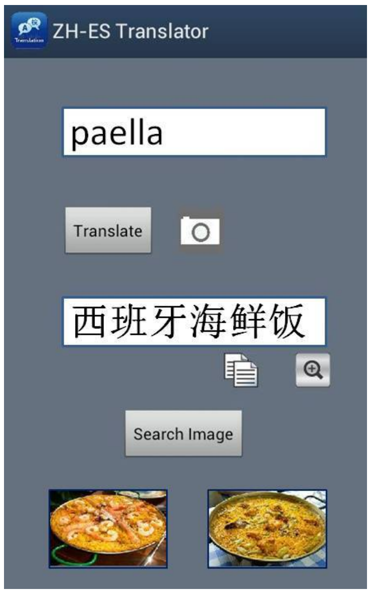


Figure 2: Android application

iOS[2]. Regarding the OCR implemented technology, this is an electronic conversion of scanned images into machine-encoded text. We adapted the open-source OCR Tesseract (released under the Apache license) (Tesseract, 2013).

3.2.2 Image retrieval

For image retrieving, we use the popular website flickr (Ludicorp, 2004). The image retrieving is activated with an specific button "search Image" button in the app (see Figure 2). Then, an URL (using the HTTPClient method) is sent to a flickr server. In the URL we specify the tag (i.e. the topic of the images we want), the number of images, the secret key (needed to interact with flickr) and also the type of object we expect (in our case, a JSON object). When the server response is received, we parse the JSON object. Afterwards, with the HTTPConnection method and the information parsed, we send the URL back to the server and we retrieve the images requested. Also, the JAVA class that implements all these methods extends an AsyncTask in order to not block the user interface meanwhile is exchanging information with the flickr servers.

3.2.3 Language detection

We have also implemented a very simple but effective language detection system, which is very suitable for distinguishing between Chinese and Spanish. Given the type of encoding we are using

[1]http://www.android.com/about/jelly-bean/

[2]http://www.apple.com/ios/siri/

(UTF-8), codes for most characters used in Spanish are in the range from 40 to 255, and codes for most characters used in Chinese are in the range from 11,000 and 30,000. Accordingly, we have designed a simple procedure which computes the average code for the sequence of characters to be translated. This average value is compared with a threshold to determine whether the given sequence of characters represents a Chinese or a Spanish input.

4 Conclusions

In this demo paper, we described "CHISPA on the GO" a translation service that allows travelers-in-need to have an easy and convenient access to Chinese-Spanish translations via a mobile app.

The main characteristics of the presented system are: the use direct translation between Chinese and Spanish, the support of both website as well as mobile platforms, and the integration of supporting input technologies such as Automatic Speech Recognition, Optical Character Recognition, Image retrieval and Language detection.

As future work we intend to exploit collected data to implement an index and search engine for providing fast access to most commonly requested translations. The objective of this enhancement is twofold: supporting off-line mode and alleviating the translation server load.

Acknowledgments

The authors would like to thank the Universitat Politècnica de Catalunya and the Institute for Infocomm Research for their support and permission to publish this research. This work has been partially funded by the Seventh Framework Program of the European Commission through the International Outgoing Fellowship Marie Curie Action (IMTraP-2011-29951) and the HLT Department of the Institute for Infocomm Reseach.

References

R. E. Banchs and H. Li. 2008. Exploring Spanish Morphology effects on Chinese-Spanish SMT. In *MATMT 2008: Mixing Approaches to Machine Translation*, pages 49–53, Donostia-San Sebastian, Spain, February.

M. R. Costa-jussà, C. A. Henríquez Q, and R. E. Banchs. 2012. Evaluating indirect strategies for chinese-spanish statistical machine translation. *J. Artif. Int. Res.*, 45(1):761–780, September.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03).*

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, pages 177–180, Prague, Czech Republic, June.

Ludicorp. 2004. Flickr. accessed online May 2013 `http://www.flickr.com/`.

F.J. Och and H. Ney. 2002. Dicriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 295–302, Philadelphia, PA, July.

L. Padró, M. Collado, S. Reese, M. Lloberes, and I. Castellón. 2010. FreeLing 2.1: Five Years of Open-Source Language Processing Tools. In *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010)*, La Valleta, Malta, May.

A. Rafalovitch and R. Dale. 2009. United Nations General Assembly Resolutions: A Six-Language Parallel Corpus. In *Proceedings of the MT Summit XII*, pages 292–299, Ottawa.

H. Saint-Amand. 2013. Moses server. accessed online May 2013 `http://www.statmt.org/moses/?n=Moses.WebTranslation`.

TausData. 2013. Taus data. accessed online May 2013 `http://www.tausdata.org`.

Tesseract. 2013. Ocr. accessed online May 2013 `https://code.google.com/p/tesseract-ocr/`.

H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*.

Appendix: Demo Script Outline

The presenter will showcase the "CHISPA on the GO" app by using the three different supported input methods: typing, speech and image. Translated results will be displayed along with related pictures of the translated items and/or locations when available. A poster will be displayed close to the demo site, which will illustrate the main architecture of the platform and will briefly explain the technology components of it.

Safe In-vehicle Dialogue Using Learned Predictions of User Utterances

Staffan Larsson
Talkamatic AB
Första Långgatan 18
413 28 Göteborg
Sweden
staffan@talkamatic.se

Fredrik Kronlid
Talkamatic AB
Första Långgatan 18
413 28 Göteborg
Sweden
fredrik@talkamatic.se

Pontus Wärnestål
Halmstad University
Box 823
301 18 Halmstad
Sweden
pontus.warnestal@hh.se

Abstract

We present a multimodal in-vehicle dialogue system which uses learned predictions of user answers to enable shorter, more efficient, and thus safer natural language dialogues.

1 Background

1.1 Driver Distraction

Driver distraction is a common cause of accidents, and is often caused by the driver interacting with technologies such as mobile phones, media players or navigation systems. A study, commonly referred to as the "100 car study" (Neale et al., 2005) revealed that secondary task distraction is the largest cause of driver inattention, and that the handling of wireless devices is the most common secondary task.

As interaction complexity in the car increases due to more advanced infotainment systems and smartphones, drivers are often executing several tasks in parallel to the primary task of driving. The increased functionality of these systems has resulted in large hierarchical information architectures that prolong interaction time, thereby negatively affecting safety as well as user experience (Kern and Schmidt, 2009).

1.2 Relation to state of the art

State-of-the-art infotainment systems typically do not include user models at all. Siri, available on the Apple iPhone 4S and later models, has a static user model containing personal information explicitly provided by the user (home address, etc.). This information is used in voice interactions; for example, given that the user has entered their family relations, phrases like "Call my wife" can be used. A different approach is taken in Google Now, which dynamically learns user patterns from observations and presents unrequested information as "cards" on the screen. However, Google Now does not attempt to integrate predictions into dialogue interaction.

The work reported here explores the use of adaptive user modeling in multimodal dialogue systems. User preferences and behaviour patterns are learnt from observations of user interactions with the infotainment system and the context in which these interactions take place, and are used proactively to predict user answers and thereby enable shorter and more efficient interaction. The underlying motivating assumption is that using apps and services in an in-vehicle context inherently leads to distraction, and that reducing interaction time will reduce driver distraction.

1.3 TDM

Based on Larsson (2002) and later work, Talkamatic AB has developed the Talkamatic Dialogue Manager (TDM).

TDM provides a general interaction model based on interaction which are basic to human-human linguistic interaction, resulting in a high degree of naturalness and flexibility which increases usability. The model is domain-independent which means that dialogue behaviour can be altered without touching application properties and vice versa. TDM also offers integrated multi-modality which allows user to freely switch between modalities (Larsson et al., 2011).

1.4 Grounding in TDM

Grounding (Clark and Brennan, 1990) is, roughly, the process of making sure that dialogue participants agree on what has been said so far and what it meant. TDM has an extensive model of grounding (Larsson, 2002). It operates on different levels:

- *Perception*
- *Semantic Understanding*

- *Pragmatic Understanding*
- *Acceptance*

System feedback (positive, negative and in some cases interrogative) can be generated on each level:

- Examples: "I didn't hear" – negative perception
- "To work, is that right?" – interrogative semantic understanding
- "OK" – positive acceptance.

2 Learning and Classification

Many dialogue applications require the user to answer a number of questions. To make dialogue shorter, we have extended TDM so that it tries to predict user answers on the basis of a user model learned from observations of user behaviour. As an illustration, we use a road information application which tries to predict the user's destination and thereby eliminate the need to ask the user about this.

2.1 Learning Method

Initially, a range of learning methods requiring (N-gram, MDP, POMDP) were explored and evaluated, but the KNN (K-Nearest Neighbours) (Mitchell, 1997) was considered the best method. An important advantage is that KNN can learn from a relatively small set of observations. This is in contrast to the MDP and POMDP (and to a lesser extent, N-gram) methods, which require large amounts of data to generate useful behaviour. A potential drawback of KNN is that this model cannot model sequences of user behaviours.

2.2 Parameter Selection

On the basis of user studies provided from the user partner of the project, it was decided that the most important user model parameters was position, day of the week and hour of the day. The training data were simulated and correspond to the behaviour of an archetypal persona provided by the user partner in the project.

2.3 Learning and Classification

The learning part of the system listens for a number of events, such as "start-car", "stop-car" etc.. From these events and information about current position, the time of the day and the day of the week, the system creates new data instances. The system thus learns how the user's destination varies depending on these parameters. A sample dataset is shown in Figure 1, where data points show destinations of trips initiated at various times of the week.

When the dialogue manager requests a prediction of the destination, the KNN algorithm tries to find the K data points closest to the present data point, and the top alternatives are returned to the dialogue manager together with confidence scores indicating the reliability of the predictions.

3 Integration of Classifications into TDM

3.1 Grounding uncertain information

We treat the information emanating from the user model as uncertain information about a (predicted) user utterance. Hence, the same mechanisms used for grounding utterances have been adapted for integrating user model data.

3.2 Integrating Classifier Output

TDM is based on the Information State Update (ISU) approach to dialogue management. The information state in TDM is based on that of the system described in Larsson (2002) and includes Questions Under Discussion, a dialogue plan, and shared commitments.

The rule for integrating the user model data is a standard ISU rule, consisting of preconditions and effects on the information state. We describe these informally below:

PRECONDITIONS

- If there is a propositional answer from the user model resolving a question in the current plan...
- and if the confidence score reported from the user model is sufficient, then...

EFFECTS

- accept the propositional answer (include it into the shared commitments), and...
- give appropriate feedback to the user depending on the confidence score:
- High confidence ⇒ embedded feedback
  - "Which route do you want to take to work?".

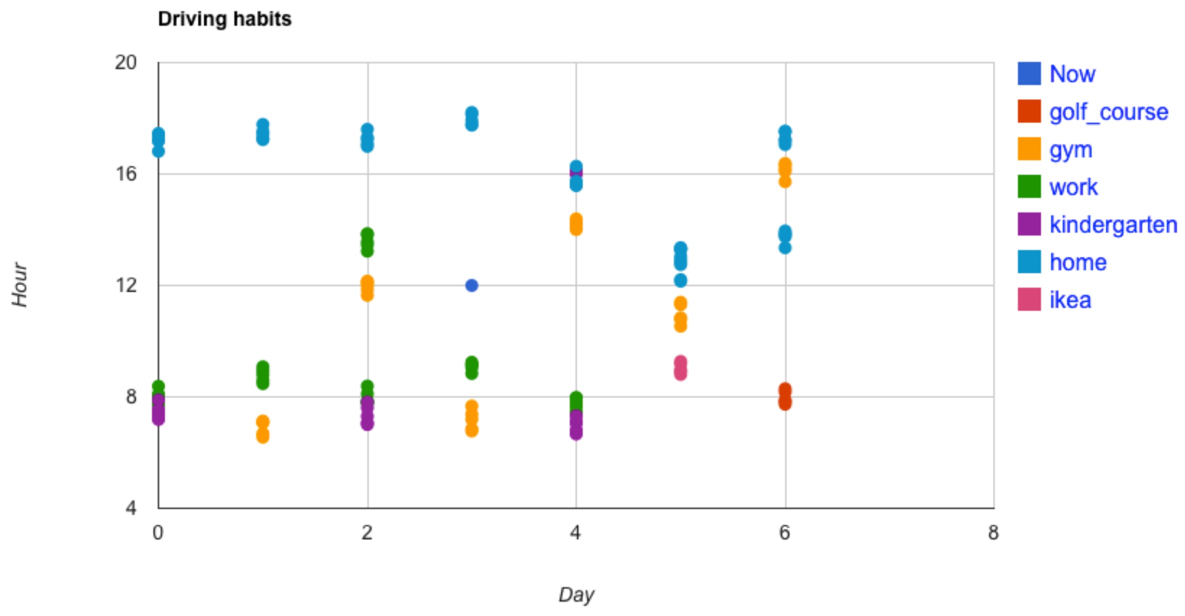


Figure 1: A sample dataset. The horizontal axis shows days of the week (0=Monday, ..., 6=Sunday) and the vertical axis shows hour of the day. Data points show destinations of trips initiated at the time indicated by their position. ("Now" is the current time, in this case Thursday at lunchtime.)

  - The user can always reject the prediction by requesting another destination.

- Medium confidence ⇒ positive feedback
  - "I assume you're going to work".
  - If the user says "no", the answer is rejected
  - Silence is interpreted as acceptance.

- Low confidence ⇒ interrogative feedback
  - "To work, is that correct?"
  - In this case, the user needs to explicitly accept the proposed answer.
  - Otherwise, the user is prompted for an answer.

3.3 GUI output

If the ISU rule above does not apply because of too low confidence scores, user model information is still used in the GUI. When a Wh-question is raised by the system, the GUI always presents a list of possible alternatives. High-confidence alternatives are highlighted and sorted before the other alternatives in the list.

4 Resulting behaviour

The demonstrator enables interaction with a learning dialogue system which uses predictions to simplify interactions. Here is an sample interaction:

```
User: Traffic information
Car: Ok. What road?
User: E6.
Car: Showing traffic on the E6
```

If this is repeated on a number of occasions, eventually the system will use a prediction:

```
User: Traffic information
Car: Showing traffic on the E6
```

The system thus reduces the need for repetitive and information-scarce utterances from the user. As soon as the system has started identifying a pattern, it will start to suggest the most probable alternatives. Initially, the most probable answers are presented to the user as the top items in a list. The alternatives are also marked in a different color to make them more visible to the user (not shown here).

```
User: Traffic information
Car: Ok. What road?
Car GUI: [E6] [E45] [E20] [155]
User: E6.
Car: Showing traffic on the E6
```

After some further use, the system has identified a pattern which is prominent enough for the system to make a suggestion:

```
User: Traffic information
Car: E6, is that right?
User: Yes.
Car: Showing traffic on the E6
```

After getting further support for its hypothesis, the system will merely inform the user that an assumption has been made. If the user is satisfied with the assumption, she does not need to do anything, but can correct or confirm it if desired.

```
User: Traffic information
Car: I assume E6.
User: [silence]
Car: Showing traffic on the E6
```

```
User: Traffic information
Car: I assume E6.
User: No, E45.
Car: Showing traffic on the E45
```

If the user rejects the system suggestion without giving another answer, the system will show a menu where the most probable choices are the topmost ones, and marked in a distinct colour (not shown here).

```
User: Traffic information
Car: I assume E6.
User: No.
Car: What road?
Car GUI: [E6] [E45] [E20] [155]
```

When the system is certain about its hypothesis, the system will simply provide the user with the desired information without asking the user for parameters.

```
User: Traffic information
Car: Showing traffic on the E6
```

5 Conclusions and further work

We have designed and implemented a mechanism which learns user patterns and uses them proactively to simplify and shorten dialogue interactions. The idea of learning user patterns from observations is similar to Google Now. However, while Google Now uses "cards" to provide unrequested information to the user, we show how predictions can be integrated into spoken or multimodal dialogue.

It remains for future work to evaluate the system to establish that this actually reduces the distraction rate of drivers. We also want to test the performance of the learning mechanism by training it on real observations of user behaviours (as opposed to simulated data).

The current mechanism only predicts answers to individual system questions, which may result in suboptimal behaviour in cases where there are dependencies between the questions pertaining to some task. An interesting area for future work is to instead predict sequences of answers; however, this would require a more powerful learning and classification mechanisms.

Acknowledgements

This work was carried out within the FFI project "Safe Speech by Knowledge" (2012-00941), funded by VINNOVA, Volvo Car Corporation and Talkamatic.

References

H. H. Clark and S. E. Brennan. 1990. Grounding in communication. In L. B. Resnick, J. Levine, and S. D. Behrend, editors, *Perspectives on Socially Shared Cognition*, pages 127 – 149. APA.

Dagmar Kern and Albrecht Schmidt. 2009. Design space for driver-based automotive user interfaces. In *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '09, pages 3–10, New York, NY, USA. ACM.

Staffan Larsson, Alexander Berman, and Jessica Villing. 2011. Adding a speech cursor to a multimodal dialogue system. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, 2011*, pages 3319–3320.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Göteborg University.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, New York.

Vicki L. Neale, Thomas A. Dingus, Sheila G. Klauer, Jeremy Sudweeks, and Michael Goodman. 2005. An overview of the 100-car naturalistic study and findings. Technical report.

Speech-Enabled Hybrid Multilingual Translation for Mobile Devices

Krasimir Angelov
University of Gothenburg
krasimir@chalmers.se

Björn Bringert
Google Inc
bringert@google.com

Aarne Ranta
University of Gothenburg
aarne@chalmers.se

Abstract

This paper presents an architecture and a prototype for speech-to-speech translation on Android devices, based on GF (Grammatical Framework). From the user's point of view, the advantage is that the system works off-line and yet has a lean size; it also gives, as a bonus, grammatical information useful for language learners. From the developer's point of view, the advantage is the open architecture that permits the customization of the system to new languages and for special purposes. Thus the architecture can be used for controlled-language-like translators that deliver very high quality, which is the traditional strength of GF. However, this paper focuses on a general-purpose system that allows arbitrary input. It covers eight languages.

1 Introduction

Many popular applications (*apps*) on mobile devices are about language. They range from general-purpose translators to tourist phrase books, dictionaries, and language learning programs. Many of the apps are commercial and based on proprietary resources and software. The mobile APIs (both Android and iOS) make it easy to build apps, and this provides an excellent way to exploit and demonstrate computational linguistics research, perhaps not used as much as it could.

GF (Grammatical Framework, (Ranta, 2011)) is a grammar formalism designed for building multilingual grammars and interfacing them with other software systems. Both multilinguality and interfacing are based on the use of an *abstract syntax*, a tree structure that captures the essence of syntax and semantics in a language-neutral way. Translation in GF is organized as *parsing* the source language input into an abstract syntax tree and then *linearizing* the tree into the target language. Here is an example of a simple question, as modelled by an abstract syntax tree and linearized to four languages, which use different syntactic structures to express the same content:

Query (What Age (Name "Madonna"))
English: *How old is Madonna?*
Finnish: *Kuinka vanha Madonna on?*
French: *Quel âge a Madonna?*
Italian: *Quanti anni ha Madonna?*

In recent years much focus in GF has been put on cloud applications (Ranta et al., 2010) and on mobile apps, for both Android (Détrez and Enache, 2010) and iOS (Djupfeldt, 2013). They all implement text-based phrasebooks, whereas Alumäe and Kaljurand (2012) have built a speech-enabled question-answering system for Estonian. An earlier speech translation system in GF is presented in Bringert (2008).

All embedded GF systems are based on a standardized run-time format of GF, called PGF (Portable Grammar Format; Angelov et al. 2009, Angelov 2011). PGF is a simple "machine language", to which the much richer GF source language is compiled by the GF grammar compiler. PGF being simple, it is relatively straightforward to write interpreters that perform parsing and linearizations with PGF grammars. The first mobile implementations were explicitly designed to work on small devices with limited resources. Thus they work fine for small grammars (with up to hundreds of rules and lexical entries per language), but they don't scale up well into open-domain grammars requiring a lexicon size of tens of thousands of lemmas. Moreover, they don't support out-of-grammar input, and have no means of choosing between alternative parse results, which in a large grammar can easily amount to thousands of trees.

A new, more efficient and robust run-time system for PGF was later written in C (Angelov, 2011). Its performance is competitive with the state of the art in grammar-based parsing (Angelov and Ljunglöf, 2014). This system uses statistical disambiguation and supports large-scale grammars, such as an English grammar covering most of the Penn Treebank. In addition, it is lean enough to be embedded as an Android application even with full-scale grammars, running even on devices as old as the Nexus One from early 2010.

Small grammars limited to natural language fragments, such as a phrasebook, are usable when equipped with predictive parsing that can suggest the next words in context. However, there is no natural device for word suggestions with speech input. The system must then require the user to learn the input language; alternatively, it can be reduced to simple keyword spotting. This can be useful in information retrieval applications, but hardly in translation. Any useful speech-enabled translator must have wide coverage, and it cannot be restricted to just translating keywords.

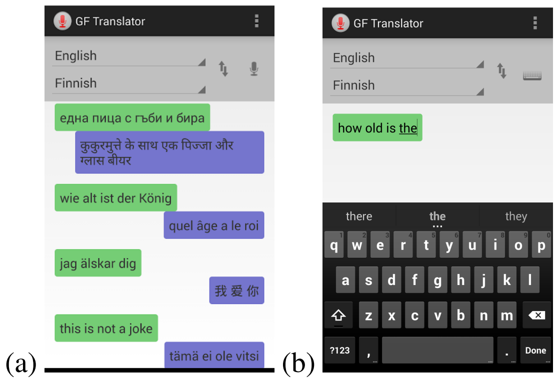
In this paper, we show a mobile system that has a wide coverage and translates both text and speech. The system is modular and could be easily adapted to traditional GF applications as well: since the PGF format is the same, one can combine any grammar with any run-time PGF interpreter.

The rest of the paper is organized as follows: Section 2 describes the system's functionalities from the user's point of view. Section 3 explains the technology from the developer's point of view. Section 4 presents some preliminary results on the usability of the system, and discusses some ways of improving it. Section 5 concludes.

A proper quantitative evaluation of the translation quality has to wait till another occasion, and will be more properly done in a context that addresses hybrid GF-based translation as a research topic. Early attempts in this area have not yet converged into a stable methodology, but we believe that setting translation in the context of a practical use case, as here, can help identify what issues to focus on.

2 Functionalities

The app starts with the last-used language pair preselected for input and output. It waits for speech input, which is invoked by touching the microphone icon. Once the input is finished, it appears in text on the left side of the screen. Its translation appears below it, on the right, and is also rendered as speech (Figure 1 (a)).


(a)

една пица с гъби и бира

कुकुरमुत्ते के साथ एक पिज्जा और ग्लास बीयर

wie alt ist der König

quel âge a le roi

jag älskar dig

我 爱 你

this is not a joke

tämä ei ole vitsi

(b)

how old is the

Figure 1: Translation between various languages with (a) speech (b) text input.

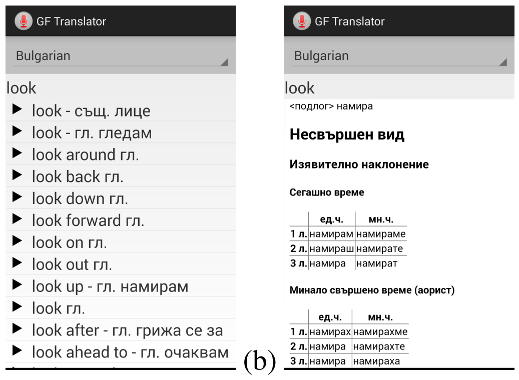
The source and target languages are selected by the two drop-down lists on the top of the screen. The icon with two arrows to the right of the language selectors allows the two languages to be swapped quickly.

The speech recognition and text-to-speech (TTS) is done using public Android APIs. On most devices, these make use of Google's speech recognizer and synthesizer, which are available in both online and offline versions. The offline engines tend to have a reduced choice of languages and reduced quality compared to the online engines, but don't require an internet connection.

Alternatively, the user can select the keyboard mode. The microphone icon is then changed to a keyboard icon, which opens a software keyboard and shows a text field for entering a new phrase. Once the phrase is translated, it is shown on the screen but also sent to TTS (Figure 1 (b)).

If the input consists of a single lexical unit, the user can open a dictionary description for the word. The resulting screen shows the base form of the word, followed by a list of possible translations. The target language is shown on the top of the screen and it can be changed to see the translations in the other languages (Figure 2 (a)). Touching one of the translations opens a full-form inflection table together with other grammatical information about the word, such as gender and verb valency (Figure 2 (b)).

Finally, the translator also works as an input mode for other apps such as SMS. It provides a soft keyboard, which is similar to the standard Android keyboard, except that it has two more keys allowing the entered phrase to be translated in-place from inside any other application.


(a)

GF Translator

Bulgarian

look

- ▶ look - същ. лице
- ▶ look - гл. гледам
- ▶ look around гл.
- ▶ look back гл.
- ▶ look down гл.
- ▶ look forward гл.
- ▶ look on гл.
- ▶ look out гл.
- ▶ look up - гл. намирам
- ▶ look гл.
- ▶ look after - гл. грижа се за
- ▶ look ahead to - гл. очаквам

(b)

GF Translator

Bulgarian

look

<подлог> намира

Несвършен вид

Изявително наклонение

Сегашно време

	ед.ч.	мн.ч.
1 л.	намирам	намираме
2 л.	намираш	намирате
3 л.	намира	намират

Минало свършено време (аорист)

	ед.ч.	мн.ч.
1 л.	намирах	намирахме
2 л.	намира	намирахте
3 л.	намира	намираха

Figure 2: (a) Results of dictionary lookup. (b) Valency and the inflection table for a Bulgarian verb.

3 Technology

3.1 Run-time processing

The core of the system is the C runtime for PGF (Angelov, 2011). The runtime is compiled to native code with the Android NDK and is called via foreign function interface from the user interface, which is implemented in Java.

The main challenge in using the runtime on mobile devices is that even the latest models are still several times slower that a modern laptop. For instance, just loading the grammars for English and Bulgarian, on a mobile device initially took about 28 seconds, while the same task is a negligible operation on a normal computer. We spent considerable time on optimizing the grammar loader and the translator in general. Now the same grammar, when loaded sequentially, takes only about 5-6 seconds. Furthermore, we made the grammar loader parallel, i.e. it loads each language in parallel. The user interface runs in yet another thread, so while the grammar is loading, the user can already start typing or uttering a sentence. In addition, we made it possible to load only those languages that are actually used, i.e. only two at a time instead of all eight at once.

Parsing is a challenge in itself. As the grammars grow bigger, there tends to be more and more need for disambiguation. This is performed by a statistical model, where each abstract syntax tree node has weight. We used the method of Angelov and Ljunglöf (2014) to find the best tree.

Moreover, since any sound grammar is likely to fail on some input, there is need for robustness. This has been solved by chunking the input into maximal parsable bits. As a result, the translations are not always grammatically correct, because dependencies between chunks, such as agreement, get lost. This kind of errors are familiar to anyone who has used a statistical system such as Google translate. In the GF system it is easy to avoid them, provided the parse is complete.

Bulgarian	26664	French	19570
Chinese	17050	German	9992
English	65009	Hindi	33841
Finnish	57036	Swedish	24550

Table 1: Lexical coverage (lemmas)

3.2 The language component

The language-specific component of the app is the PGF grammar, which contains both the grammars proper and the probabilistic model of the abstract syntax. The app can be adaptad to a different PGF grammar by changing a few lines of the source code. Hence any grammar written in GF is readily usable as the language component of an app. But here we focus on the large-scale grammar meant for robust translation.

The core of the grammar is the GF Resource Grammar Library (Ranta, 2009), which currently covers 29 languages. Of these, 8 have been extended with more syntax rules (about 20% in addition to the standard library) and a larger lexicon. Table 1 shows the list of languages together with the size of the lexicon for each of them. The abstract syntax is based on English lemmas and some split word senses of them. The other languages, having fewer words than English, are thus incomplete. Unknown words are rendered by either showing them in English (if included in the English lexicon) or just returning them verbatim (typical for named entities).

The lexicon has been bootstrapped from various freely available sources, such as linked WordNets and the Wiktionary. Parts of the lexicon have been checked or completely written manually.

4 First results

The most striking advantage of the translation app is its lean size: currently just 18Mb for the whole set of 8 languages, allowing translation for 56 language pairs. This can be compared with the size of about 200Mb for just one language pair in Google's translation app used off-line. The Apertium off-line app is between these two, using around 2MB per language pair.

The speed is still an issue. While the app now loads smoothly on modern hardware (such as Nexus 5 phones), translation is usually much slower than in Google and Apertium apps. The speed depends heavily on the complexity of the source language, with Finnish and French the worst ones, and on sentence length. Only with short sentences (under ten words) from Bulgarian, Chinese, English, and Swedish, does the translator deliver satisfactory speed. On the other hand, long sentences entered via speech are likely to contain speech recognition errors, which makes their translation pointless anyway.

Translating single words is based on a simpler algorithm (dictionary lookup) and is therefore immediate; together with the grammatical information displayed, this makes single word translation into the most mature feature of the app so far.

The translation quality and coverage are reasonable in phrasebook-like short and simple sentences. The app has exploited some idiomatic constructions of the earlier GF phrasebook (Détrez and Enache, 2010), so that it can correctly switch the syntactic structure and translate e.g. *how old are you* to French as *quel âge as-tu*. In many other cases, the results are unidiomatic word-to-word translations but still grammatical. For instance, *hur mycket är klockan*, which should give *what is the time*, returns *how mighty is the bell*. Such short idioms are typically correct in Google's translation app, and collecting them into the GF resources will be an important future task.

On the plus side, grammar-based translation is more predictable than statistical. Thus (currently) when using Google translate from Swedish to English, both *min far är svensk* and its negation *min far är inte svensk* come out as the positive sentence *my father is Swedish*. With grammar-based translation, such semantic errors can be avoided.

5 Conclusion

We have presented a platform for mobile translation apps based on GF grammars, statistical disambiguation, and chunking-based robustness, enhanced by Android's off-the-shelf speech input and output. The platform is demonstrated by a system that translates fairly open text between 8 languages, with reasonable performance for short sentences but slow parsing for longer ones, with moreover lower quality due to more parse errors.

The processing modules, user interface, and the language resources are available as open source software and thereby usable for the community for building other systems with similar functionalities. As the app is a front end to a grammatical language resource, it can also be used for other language-aware tasks such as learning apps; this is illustrated in the demo app by the display of inflection tables. The app and its sources are available via http://www.grammaticalframework.org.

References

Tanel Alumäe and Kaarel Kaljurand. 2012. Open and extendable speech recognition application architecture for mobile environments. *The Third International Workshop on Spoken Languages Technologies for Under-resourced Languages (SLTU 2012), Cape Town, South Africa.*

Krasimir Angelov and Peter Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *European Chapter of the Association for Computational Linguistics*, Gothenburg.

Krasimir Angelov, Björn Bringert, and Aarne Ranta. 2009. PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information*, 19(2), pp. 201–228.

Krasimir Angelov. 2011. *The Mechanics of the Grammatical Framework.* Ph.D. thesis, Chalmers University of Technology.

Björn Bringert. 2008. Speech translation with Grammatical Framework. In *Coling 2008: Proceedings of the workshop on Speech Processing for Safety Critical Translation and Pervasive Applications*, pages 5–8, Manchester, UK, August. Coling 2008 Organizing Committee.

Grégoire Détrez and Ramona Enache. 2010. A framework for multilingual applications on the android platform. In *Swedish Language Technology Conference.*

Emil Djupfeldt. 2013. Grammatical framework on the iphone using a C++ PGF parser. Technical report, Chalmers Univerity of Technology.

Aarne Ranta, Krasimir Angelov, and Thomas Hallgren. 2010. Tools for multilingual grammar-based translation on the web. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 66–71, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aarne Ranta. 2009. The GF resource grammar library. *Linguistic Issues in Language Technology.*

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars.* CSLI Publications, Stanford. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

The New THOT Toolkit for Fully-Automatic and Interactive Statistical Machine Translation

Daniel Ortiz-Martínez
Dpto. de Sist. Inf. y Comp.
Univ. Politéc. de Valencia
46071 Valencia, Spain
dortiz@dsic.upv.es

Francisco Casacuberta
Dpto. de Sist. Inf. y Comp.
Univ. Politéc. de Valencia
46071 Valencia, Spain
fcn@dsic.upv.es

Abstract

We present the new THOT toolkit for fully-automatic and interactive statistical machine translation (SMT). Initial public versions of THOT date back to 2005 and did only include estimation of phrase-based models. By contrast, the new version offers several new features that had not been previously incorporated. The key innovations provided by the toolkit are computer-aided translation, including post-editing and interactive SMT, incremental learning and robust generation of alignments at phrase level. In addition to this, the toolkit also provides standard SMT features such as fully-automatic translation, scalable and parallel algorithms for model training, client-server implementation of the translation functionality, etc. The toolkit can be compiled in Unix-like and Windows platforms and it is released under the GNU Lesser General Public License (LGPL).

1 Introduction

Open-source software constitutes a valuable resource for researchers or companies. Due to the inherent difficulties of developing good quality software (correct, efficient, modular, extensible, well-documented, etc.), there are interesting research ideas that not always receive enough attention from the open-source software community.

We present the THOT toolkit for statistical machine translation (SMT). The first public version of THOT was initially created in 2005 (Ortiz et al., 2005) and its functionality was restricted to train phrase-based models (Koehn et al., 2003). Here we present a new version of THOT which includes several new features related to phrase-based translation. More specifically, the set of features provided by THOT can be classified into advanced features and standard features. Advanced features correspond to sophisticated functionality that has received poor or no attention in existing SMT toolkits. By contrast, standard features correspond to functionality already provided by popular tools such as Moses (Koehn et al., 2007). In this regard, THOT neither is based on Moses nor shares any source code with it.

THOT includes the following advanced features:

- Computer-aided translation, including post-editing and interactive machine translation (IMT). This functionality has been integrated in a translation tool developed in the CasMaCat project[1] (the so-called CasMaCat Workbench).
- Incremental estimation of all of the models involved in the translation process.
- Robust generation of phrase-based alignments.

Computer-aided translation and more specifically two of its applications, post-editing and IMT, constitute a field of increasing interest in SMT. In particular, IMT has been studied in numerous research papers during the last years. In spite of this, this application has not previously been implemented in open-source software tools.

Incremental (or online) learning is a hot research topic in SMT due to the great interest of quickly incorporating incoming data into existing translation systems. In spite of the fact that the Moses toolkit already implements incremental learning techniques, such techniques are designed to work by incrementally processing large blocks of data and not in a sentence-wise manner, as it is pointed out in (Mirking and Cancedda, 2013). By

[1] http://www.casmacat.eu/

contrast, the incremental learning techniques implemented by THOT allows to process new training samples individually in real time.

Finally, the necessity of generating phrase-level alignments is present in a wide range of tasks, from multisource SMT to discriminative training. However, as far as we know this functionality also is not included in existing SMT tools.

In addition to the above mentioned advanced features, THOT offers a set of standard features:

- Phrase-based SMT decoder.
- Scalable training and search algorithms.
- Client-server implementation.
- Miscellaneous SMT tools

2 The THOT toolkit

THOT can be downloaded from GitHub[2] and is distributed under the GNU Lesser General Public License (LGPL). It has been developed using C++ and shell scripting. The design principles that have led the development process were:

- **Modularity**: The THOT code is organised into separate packages for each main functional component (training of phrase-based and language models, decoding, etc.). Each component can be treated as a stand-alone tool and does not rely on the rest of the code.
- **Extensibility**: The functionality provided by each package is structured into classes. Abstract classes are used when appropriate to define the basic behaviour of the functional components of the toolkit, allowing us to easily extend the toolkit functionality.
- **Scalability**: THOT is able to train statistical models from corpora of an arbitrary size. Moreover, the toolkit takes advantage of parallel and distributed computing to reduce the time cost of the implemented algorithms. Additionally, the parameters of the resulting models can be pruned or accessed from disk during the decoding process.
- **Portability**: It is known to compile on Unix-like and Windows (using Cygwin) systems.

In the rest of the paper we give additional details about the different toolkit features that have been mentioned above.

[2]`https://github.com/daormar/thot`

3 Computer-Aided Translation

Current MT systems are not able to produce ready-to-use texts. Indeed, they usually require human post-editing in order to achieve high-quality translations. This motivates an alternative application of MT in which the MT system collaborates with the user to generate the output translations. This alternative application receives the name of computer-assisted translation (CAT).

CAT can be instantiated in different ways. The THOT toolkit incorporates tools that are useful in two different CAT instantations, namely, post-editing and interactive machine translation.

3.1 Post-Editing

Post-editing (PE) involves making corrections and amendments to machine generated translations (see (TAUS, 2010) for a detailed study). In the PE scenario, the user only edits the output of the MT system without further system intervention.

3.2 Interactive Machine Translation

In the IMT framework (Foster et al., 1997; Langlais et al., 2002), the user obtains her desired translations in a series of interactions with an MT system. Specifically, the system initially generates a translation without human intervention and after that, the user validates a prefix of the translation and introduce the next correct character of it. With this information, the IMT system returns the suffix which best completes the user prefix. This process is repeated until the user gets the sentence she has in mind. In (Barrachina et al., 2009), SMT techniques were embedded within the interactive translation environment.

A common problem in IMT arises when the user sets a prefix which cannot be explained by the statistical models. This problem requires the introduction of specific techniques to guarantee that the suffixes can be generated. The majority of the IMT systems described in the literature use error-correcting techniques based on the concept of edit distance to solve the coverage problems. Such error-correction techniques, although they are not included in the statistical formulation of the IMT process, are crucial to ensure that the suffixes completing the user prefixes can be generated.

THOT implements an alternative formalisation that introduces stochastic error-correction models in the IMT statistical formulation. Such a formalisation was introduced in (Ortiz-Martínez, 2011) and it generates the suffixes required in IMT by partially aligning a prefix of the target hypotheses with the user prefix. Once the partial alignment is determined, the suffix is given by the unaligned portion of the target sentence.

Experiments to test the above mentioned IMT proposal were carried out using THOT. The results showed that the proposed IMT system outperforms the results of other state-of-the-start IMT systems that are based on word graphs (see (Ortiz-Martínez, 2011) for more details).

3.3 Integration with the CasMaCat Workbench

THOT can be combined with the CasMaCat Workbench[3] that is being developed within the project of the same name. The CasMaCat Workbench offers novel types of assistance for human translators, using advanced computer aided translation technology that includes PE and IMT.

4 Incremental Learning for SMT

Thot incorporates techniques to incrementally update the parameters of the statistical models involved in the translation process. Model updates can be quickly executed in a sentence-wise manner allowing the system to be used in a real time scenario. For this purpose, a log-linear SMT model where all its score components are incrementally updateable is defined. The implemented proposal uses the incremental version of the EM algorithm (Neal and Hinton, 1998) and the specific details can be found in (Ortiz-Martínez et al., 2010; Ortiz-Martínez, 2011).

Empirical results obtained with THOT and reported in (Ortiz-Martínez et al., 2010; Ortiz-Martínez, 2011) show that incremental learning allows to significantly reduce the user effort in IMT tasks with respect to that required by a conventional IMT system.

Additionally, the incremental learning techniques provided by THOT are currently being used in other sophisticated applications such as active learning for SMT (González-Rubio et al., 2012).

5 Generation of Phrase-Based Alignments

The generation of phrase-level alignments is interesting due to its utility in a wide range of applications, including multi-source SMT, Viterbi-like estimation of phrase-based models or discriminative training, just to name a few.

A very straightforward technique can be proposed for finding the best phrase-alignment. Specifically, the search process only requires a regular SMT system which filters its phrase table in order to obtain those target translations for the source sentence that are compatible with the given target sentence. Unfortunately, this technique has no practical interest when applied on regular tasks due to problems with unseen events.

To overcome the above-mentioned difficulty, an alternative technique that is able to consider every source phrase of the source sentence as a possible translation of every target phrase of the target sentence can be defined. The THOT toolkit implements the proposal described in (Ortiz-Martínez et al., 2008), which combines a specific search algorithm with smoothing techniques to enable efficient exploration of the set of possible phrase-alignments for a sentence pair.

Phrase-based alignment quality was difficult to evaluate since there is not a gold standard for this task. One way to solve this problem consists in refining the phrase alignments to word alignments and compare them with those obtained in existing shared tasks on word alignment evaluation. Results obtained with THOT reported in (Ortiz-Martínez et al., 2008) clearly show the efficacy of the implemented method.

6 Standard Features

THOT incorporates a number of standard features that are present in existing translation tools. Such standard features are briefly enumerated and described in the following paragraphs.

Phrase-Based SMT Decoder The toolkit implements a state-of-the-art phrase-based SMT decoder. The decoder uses a log-linear model with a complete set of components similar to those implemented in other tools such as Moses. Results reported in (Ortiz-Martínez, 2011) show that the translation quality obtained by THOT is comparable to that obtained by means of Moses.

Scalable Training and Search Algorithms Due to the increasing availability of large training corpora, it is necessary to implement scalable training and search algorithms. THOT incorporates tools to train statistical models from corpora

[3]See installation instructions at `http://www.casmacat.eu/index.php?n=Workbench.Workbench`

of an arbitrary size. Such tools can take advantage of the availability of multiple processors or computer clusters. The parameters of the resulting models can be pruned or accessed from disk during the decoding stage.

Client-Server Implementation An important part of the functionality provided by the toolkit can be accessed using a client-server model. This is a useful feature to build web applications offering SMT services.

Miscellaneous SMT tools THOT reduces dependencies with third-party software by integrating most critical components of a typical machine translation pipeline, from the estimation of phrase-based and language models to the generation of translations and their automatic evaluation. The estimation of word-alignment models using the incremental EM algorithm is also implemented by the toolkit.

7 Conclusions

THOT is an open-source toolkit for SMT designed for its use in Unix-like and Windows systems. It has been developed using C++ and shell scripting, and it is released under LGPL license. THOT incorporates three advanced features that have received little attention in previous publicly-available SMT tools, namely, interactive machine translation, incremental learning and generation of phrase-based alignments. Additionally, THOT also implements standard features such as training of statistical models or decoding. The functionality of the toolkit has been empirically tested, showing its efficacy in different SMT-related tasks.

Acknowledgments

Work supported by the European Union 7th Framework Program (FP7/2007-2013) under the CasMaCat project (grant agreement nº 287576), by Spanish MICINN under grant TIN2012-31723, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

References

S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. L. Lagarda, H. Ney, J. Tomás, E. Vidal, and J. M. Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

G. Foster, P. Isabelle, and P. Plamondon. 1997. Target-text mediated interactive machine translation. *Machine Translation*, 12(1):175–194.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2012. Active learning for interactive machine translation. In *Procs. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–254.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Procs. of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 48–54, Edmonton, Canada, May.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Procs. of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic, June.

P. Langlais, G. Lapalme, and M. Loranger. 2002. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation*, 15(4):77–98.

S. Mirking and N. Cancedda. 2013. Assessing quick update methods of statistical translation models. In *Procs. of International Workshop of Spoken Language Translation*, pages 264–271, Heidelberg, Germany.

R.M. Neal and G.E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Procs. of the NATO-ASI on Learning in graphical models*, pages 355–368, Norwell, MA, USA.

D. Ortiz, I. García-Varea, and F. Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. In *Machine Translation Summit*, pages 141–148, Phuket, Thailand, September.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2008. Phrase-level alignment generation using a smoothed loglinear phrase-based statistical alignment model. In *Procs. of the European Association for Machine Translation*.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Procs. of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 546–554.

D. Ortiz-Martínez. 2011. *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. Ph.D. thesis, Universidad Politécnica de Valencia.

TAUS. 2010. Postediting in practice. a TAUS report. Technical report, March.

A Lightweight Terminology Verification Service for External Machine Translation Engines

Alessio Bosca[†], **Vassilina Nikoulina**[‡], **Marc Dymetman**[‡]
[†]CELI, Turin, Italy
[‡]Xerox Research Centre Europe, Grenoble, France
[†]alessio.bosca@celi.it, [‡]{*first.last*}@xrce.xerox.com

Abstract

We propose a demonstration of a domain-specific terminology checking service which works on top of any generic black-box MT, and only requires access to a bilingual terminology resource in the domain. In cases where an incorrect translation of a source term was proposed by the generic MT service, our service locates the wrong translation of the term in the target and suggests a terminologically correct translation for this term.

1 Introduction

Today there exist generic MT services for a large number of language pairs, which allow relatively easily to make your domain-specific portal multilingual, and allow access to its documents for a broad international public. However, applying a generic MT service to domain-specific texts often leads to wrong results, especially relative to the translation of domain-specific terminology. Table 1 illustrates an example of a terminology inconsistent translation provided by a generic MT system.

English Source: Farmers tend to implement a broad non-focused **weed-control** strategy, on the basis of broad spectrum products and mixtures of different products.
Bing[1]: Los agricultores tienden a aplicar una estrategia amplia para **control de malezas** no centrado, sobre la base de productos de amplio espectro y las mezclas de diferentes productos.

Table 1: Example of the translation produced by a generic MT model for a domain-specific document. Source term : **weed-control**, official Spanish term translation: **control de malas hierbas**.

The importance of domain-specific terminology for Machine Translation has been mentioned in several previous works (eg. (Carl and Langlais, 2002; Skadins et al., 2013)). However, most of these works handle the case where the terminology is tightly integrated into the translation process. This requires both a good expertise in SMT and a large amount of both in-domain and generic parallel texts, which is often difficult, especially for low-resourced languages like Turkish or Estonian. Here, we are targeting the situation where the content provider is not willing to train a dedicated translation system, for some reason such as lack of technical skills or lack of necessary resources (parallel data or computational resources), but has at his disposal a multilingual in-domain terminology which could be helpful for improving the generic translation provided by an external translation service. We propose a demonstration of a multilingual terminology verification/correction service, which detects the wrongly translated terms and suggests a better translation of these terms. This service can be seen as an aid for machine translation post-editing focused on in-domain terminology and as a tool for supporting the workflow of practicing translators.

2 Related Work

There has recently been a growing interest for terminology integration into MT models. Direct integration of terminology into the SMT model has been considered, either by extending SMT training data (Carl and Langlais, 2002), or via adding an additional term indicator feature (Pinnis and Skadins, 2012; Skadins et al., 2013) into the translation model. However none of the above is possible when we deal with an external black-box MT service.

(Itagaki and Aikawa, 2008) propose a post-processing step for an MT engine, where a wrongly translated term is replaced with a user-provided term translation. The authors claim that translating the term directly often gives a different translation from the one obtained when translating the term in context: for English-Japanese the out-of-context term translation matches exactly the in context term translation in 62% of cases only. In order to address this problem the authors propose 15 simple context templates that induce the same term translation as the one obtained in the initial sentence context. Such templates include ”This is TERM” or ”TERM is a ...”. The main problem with this approach is that these templates are both language-pair and MT engine/model specific. Thus a certain human expertise is required to develop such templates when moving to a new language pair or underlying MT engine.

Our approach is close to the (Itagaki and Aikawa, 2008) approach, but instead of developing specific templates we propose a generic method for wrong terminology translation detection. We do not aim at producing the final translation by directly replacing the wrongly translated term — which can be tricky—, but rather perform the term correction in an interactive manner, where the user is proposed a better term translation and may choose to use it if the suggestion is correct.

3 Terminology-checking service

We assume that the provider of the terminology-checking service has a bilingual domain-specific terminology D at his disposal, which he wishes to use to improve the translation produced by a generic MT service MT. Our method verifies whether the terminology was translated correctly by the MT service (terminology verification), and if not, locates the wrong translation of the term and suggests a better translation for it.

3.1 Terminology checking

The basic terminology verification procedure applied to the source sentence s and to its translation $MT(s)$ by the generic service is done through the following steps:

1. For each term $T = (T_s, T_t)$ in D check whether its source part T_s is present in the source sentence s.

2. If s contains T_s, check whether the target part of the term T_t is present in the translation $MT(s)$. If yes, and the number of occurrences of T_s in s is equal to that of T_t in $MT(s)$: the term translation is consistent with terminological base. Otherwise, we attempt to locate the wrong term translation and suggest a better translation to the user.

Both steps require a sub-string matching algorithm which is able to deal with term detection problems such as morphological variants or different term variants. We describe the approach we take for efficient sub-string matching in more detail in section 3.3.

3.2 Terminology correction

Once we have detected that there is a source term T_s which has been incorrectly translated we would like to suggest a better translation for this term. This requires not only knowing a correct translation T_t of the source term T_s, but also its position in the target sentence. To do that, we need to identify what was the incorrect translation proposed by the MT engine for the term and to locate it in the translation $MT(s)$.

This can be seen as a sub-problem of the word-alignment problem, which is usually solved using bilingual dictionaries or by learning statistical alignment models out of bilingual corpora. However, in practice, these resources are not easily available, especially for low-resourced language pairs. In order to be able to locate the wrong term translation in the target sentence without resorting to such resources, our approach is to rely instead on the *same* external MT engine that was used for translating the whole source sentence in the first place, an approach also taken in (Itagaki and Aikawa, 2008).

To overcome the problem mentioned by (Itagaki and Aikawa, 2008) of non-matching out-of-context terms translations we propose to combine out-of context term translation ($MT(T_s)$) and context-extended term translation, as follows:

- Translate the term T_s extended with its left and/or right n-gram context: $s_{i-n}s_{i-n+1}...T_s...s_{j+n-1}s_{j+n}$, where $T_s = s_i...s_j$;

- Find a fuzzy match in $MT(s)$ for the translation of the context-extended term $MT(s_{i-n}...T_s...s_{j+n})$ using the same sub-string matching algorithm as in the terminology verification step.

Various combinations of out-of-context term translation ($MT(T_s)$) and n-extended term translation ($MT(s_{i-n}...T_s...s_{j+n})$) are possible.

The term location is performed in a sequential way: if the wrong term translation was not located after the first step (out-of-context translation), attempt the following step, extending size of the context (n) until the term is located.

3.3 Implementation

The implementation of the terminology-checking service that we demonstrate exploits Bing Translator[2] as SMT service, refers to the Agriculture domain and supports two terminology resources: the multilingual ontology from the Organic.Edunet portal[3] and Agrovoc, a multilingual theasurus from FAO[4]. The presented prototype enables terminology checking for all the language pairs involving English, French, German, Italian, Portoguese, Spanish and Turkish.

The component for matching the textual input (i.e. either the source or the translation from the SMT service) with elements from domain terminologies is based on the open source search engine Lucene[5] and exploits its built-in textual search capabilities and indexing facilities. We created a search index for each of the supported languages, containing the textual representations of the terminology elements in that language along with their URI (unique for each terminology element). The terms expressions are indexed in their original form as well as in their lemmatized and POS tagged ones; for Turkish, resources for morphological analysis were not available therefore stemming has been used instead of lemmatization.

In order to find the terminological entries within a textual input in a given language a two-steps procedure is applied:

- In a first step, the text is used as a query over the search index (in that language) in order to find a list of all the terminology elements containing a textual fragment present in the query.

- In a second step, in order to retain only the domain terms with a complete match (no partial matches) and locate them in the text, a new search index is built in memory, containing a single document, namely the original textual input (lemmatized or stemmed according to the resources available for that specific language). Then the candidate terminology elements found in the first step are used as queries over the in-memory index and the “highlighter” component of the search engine is exploited to locate them in the text (when found). A longest match criterion is used when the terminology elements found refer to overlapping spans of text.

Following this procedure a list with terminology elements (along with their URIs and the position within the text) is generated for both the source text and its translation. A matching strategy based on the URI allows to pair domain terms from the two collections. For domain terms in the source text without a corresponding terminology element in the translated text, the ”wrong” translation is located in the text according to the approach described in 3.2. The domain term is retranslated with the same SMT (with context extension, if needed) in order to obtain the ”wrong” translation and the translated string is located within the translation text with the same approach used in the second step of the procedure used for locating terminological entries (with an in-memory search index over the full text and the fragment used as query).

The service outputs two lists: one containing the pairs of terminology elements found both in the source and in the translation and another one with the terminology elements without a ”correct” translation (according to the domain terminology used) and for each of those an alternative translation from the domain terminology is proposed. In our demonstration a web interface allows users to access and test the service.

4 Proof of concept evaluation

In order to evaluate the quality of locating the wrong term translation, we applied the terminology verification service to an SMT model trained with Moses (Hoang et al., 2007) on the Europarl (Koehn, 2005) corpus. This SMT model was used for translating a test set in the Agricultural domain from Spanish into English. In these settings we have access to the internal sub-phrase alignment provided by Moses, thus we know the exact location of the wrong term translation, which allows us to evaluate how good our locating technique is.

The test set consists of 100 abstracts in Spanish from a bibliographical database of scientific publications in the Agriculture domain. These abstracts were translated into English with our translation

[2]http://www.bing.com/translator
[3]http://organic-edunet.eu/
[4]http://aims.fao.org/standards/agrovoc/about
[5]https://lucene.apache.org

model, and we then applied terminology verification and terminology correction procedures to these translations.

When applying terminology verification we detected in total 171 terms in Spanish, 71 of them being correctly translated into English (consistent with terminology), and 100 being wrongly translated (not consistent with terminology).

We then attempted to locate these wrongly translated terms in the system translation $MT(s)$.

Matching the out-of-context term translation with initial translation allowed to find a match for 82 wrongly translated terms (out of 100); Matching 1 left/right word extended term translation $(MT(w_{i-1}T_s w_{j+1}))$ allowed to find a match for 16 more terms (out of 18 left).

Using the internal word alignments provided by Moses, we also evaluated how precisely the borders of the wrongly translated term were recovered by our term location procedure. This precision is measured as follows:

- The target tokens identified by our procedure (as described in 3) are: $g_T = t_1, \ldots, t_j$;
- We then identify the reference target tokens corresponding to the translation of the term T_s using the Moses word alignment : $r_T = \{r_{t_1}, \ldots, r_{t_k}\}$.

We define term location precision p as $p = \frac{|t_j \in r_T \cap g_T|}{|g_T|}$. The precision of term location with out-of-context term translation is of 0.92; the precision of term location with context-extended term translation is 0.91.

Overall, our approach allows to match 98% of the wrongly translated terms, with an overall location precision of 0.91. Although these numbers may vary for other language pairs and other MT systems, this performance is encouraging.

5 Conclusion

We propose a demonstration of a terminology verification system that can be used as an aid for post-editing machine translations explicitly focused on bilingual terminology consistency. This system relies on an external black-box generic MT engine extended with available domain-specific terminology. The location of the wrong term translation is located via re-translation of the original term with the same MT engine. We show that we partially overcome the situation where the out-of-context translation of the term differs from the original translation of this term (in the full sentence) by extending the term context with surrounding n-grams. The terminology verification method is both MT engine and language independent, does not require any access to the internals of the MT engine used, and is easily portable.

Acknowledgments

This work has been partially supported by the Organic.Lingua project (http://www.organic-lingua.eu/), funded by the European Commission under the ICT Policy Support Programme.

References

Michael Carl and Philippe Langlais. 2002. An intelligent terminology database as a pre-processor for statistical machine translation. In *COLING-02: Second International Workshop on Computational Terminology*, pages 1–7.

Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondej Bojar. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007 Demo and Poster Sessions*, pages 177–180.

Masaki Itagaki and Takako Aikawa. 2008. Post-mt term swapper: Supplementing a statistical machine translation system with a user dictionary. In *LREC*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit X*, pages 79–86, Phuket Thailand.

Marcis Pinnis and Raivis Skadins. 2012. Mt adaptation for under-resourced domains - what works and what not. In *Baltic HLT*, volume 247, pages 176–184.

Raivis Skadins, Marcis Pinnis, Tatiana Gornostay, and Andrejs Vasiljevs. 2013. Application of online terminology services in statistical machine translation. In *MT Summit XIV*, pages 281–286.

Finding Terms in Corpora for Many Languages with the Sketch Engine

Adam Kilgarriff
Lexical Computing Ltd., United Kingdom
adam.kilgarriff@sketchengine.co.uk

Miloš Jakubíček and **Vojtěch Kovář** and **Pavel Rychlý** and **Vít Suchomel**
Masaryk University, Czech Republic
Lexical Computing Ltd., United Kingdom
{xjakub, xkovar3, pary, xsuchom2}@fi.muni.cz

1 Overview

Term candidates for a domain, in a language, can be found by

- taking a corpus for the domain, and a reference corpus for the language
- identifying the grammatical shape of a term in the language
- tokenising, lemmatising and POS-tagging both corpora
- identifying (and counting) the items in each corpus which match the grammatical shape
- for each item in the domain corpus, comparing its frequency with its frequency in the refence corpus.

Then, the items with the highest frequency in the domain corpus in comparison to the reference corpus will be the top term candidates.

None of the steps above are unusual or innovative for NLP (see, e. g., (Aker et al., 2013), (Gojun et al., 2012)). However it is far from trivial to implement them all, for numerous languages, in an environment that makes it easy for non-programmers to find the terms in a domain. This is what we have done in the Sketch Engine (Kilgarriff et al., 2004), and will demonstrate. In this abstract we describe how we addressed each of the stages above.

2 The reference corpus

Lexical Computing Ltd. (LCL) has been building reference corpora for over a decade. Corpora are available for, currently, sixty languages. They were collected by LCL from the web. For the world's major languages (and some others), these are in the billions of words, gathered using SpiderLing (Suchomel and Pomikálek, 2012) and forming the TenTen corpus family (Jakubíček et al., 2013).

3 The domain corpus

There are two situations: either the user already has a corpus for the domain they are interested in, or they do not. In the first case, there is a web interface for uploading and indexing the corpus in the Sketch Engine. In the second, we offer WebBootCaT (Baroni et al., 2006), a procedure for sending queries of 'seed terms' to a commercial search engine; gathering the pages that the search engine identifies; and cleaning, deduplicating and indexing them as a corpus (Baroni and Bernardini, 2004). (The question "how well does it work?" is not easy to answer, but anecdotal evidence over ten years suggests: remarkably well.)

4 Grammatical shape

We make the simplifying assumption that terms are noun phrases (in their canonical form, without leading articles: the term is *base station*, not *the base stations*.) Then the task is to write a noun phrase grammar for the language.

5 Tokenising, lemmatising, POS-tagging

For each language, we need processing tools. While many in the NLP world make the case for language-independent tools, and claim that their tools are usable for any, or at least many, languages, we are firm believers in the maxim "never trust NLP tools from people who don't speak the language". While we use language-independent components in some cases (in particular TreeTagger,[1] RFTagger[2] and FreeLing[3]), we collaborate with NLP experts in the language to ascertain what the best available tools are, sometimes to assist

[1]http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
[2]http://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/
[3]http://nlp.lsi.upc.edu/freeling/

in obtaining and customising them, and to verify that they are producing good quality output. In most cases these collaborators are also the people who have written the sketch grammar and the term grammar for the language.[4]

6 Identifying and counting candidates

Within the Sketch Engine we already have machinery for shallow parsing, based on a 'Sketch Grammar' of regular expressions over part-of-speech tags, written in CQL (Corpus Query Language, an extended version of the formalism developed in Stuttgart in the 1990s (Schulze and Christ, 1996)). Our implementation is mature, stable and fast, processing million-word corpora in seconds and billion-word corpora in a few hours.

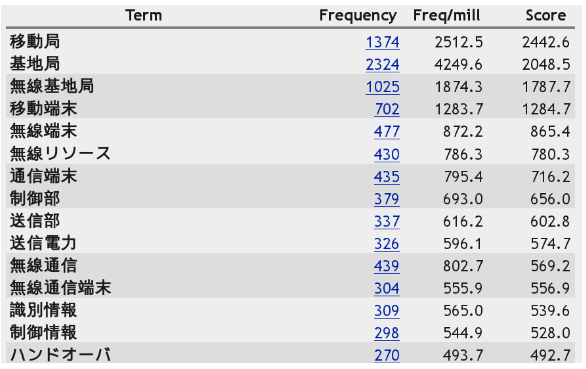
The machinery has most often been used to find <grammatical-relation, word1, word2> triples for lexicography and related research. It was straightforward to modify it to find, and count, the items having the appropriate shape for a term.

7 Comparing frequencies

The challenge of identifying the best candidate terms for the domain, given their frequency in the domain corpus and the reference corpus, is a variant on the challenge of finding the keywords in a corpus. As argued in (Kilgarriff, 2009), a good method is simply to take the ratio of the normalised frequency of the term in the domain corpus to its normalised frequency in a reference corpus. Before taking the ratio, we add a constant, the 'simple maths parameter', firstly, to address the case where the candidate is absent in the reference corpus (and we cannot divide by zero), and secondly, because there is no one right answer: depending on the user needs and on the nature of the corpora, the constant can be raised to give a list with more higher-frequency candidates, or lowered to give more emphasis to lower-frequency items.

Candidate terms are then presented to the user in a sorted list, with the best candidates – those with the highest domain:reference ratio – at the top. Each item in the list is clickable: the user can click to see a concordance for the term, in either the domain or the reference corpus.

[4]Collaborators are typically credited on the 'info' page for a reference corpus on the Sketch Engine website. The collaborations are also often agreeable and fruitful in research terms, resulting in many joint publications.



Term	Frequency	Freq/mill	Score
移動局	1374	2512.5	2442.6
基地局	2324	4249.6	2048.5
無線基地局	1025	1874.3	1787.7
移動端末	702	1283.7	1284.7
無線端末	477	872.2	865.4
無線リソース	430	786.3	780.3
通信端末	435	795.4	716.2
制御部	379	693.0	656.0
送信部	337	616.2	602.8
送信電力	326	596.1	574.7
無線通信	439	802.7	569.2
無線通信端末	304	555.9	556.9
識別情報	309	565.0	539.6
制御情報	298	544.9	528.0
ハンドオーバ	270	493.7	492.7

Figure 2: Term finding results for Japanese, WIPO format.

8 Current status

Languages currently covered by the terminology finding system are sumarized in Table 1.

Language	POS tagger	Ref. corpus
Chinese simp.	Stanford NLP	zhTenTen11
Chinese trad.	Stanford NLP	zhTenTen11
English	TreeTagger	enTenTen08
French	TreeTagger	frTenTen12
German	RFTagger	deTenTen10
Japanese	MeCab+Comainu	jpTenTen11
Korean	HanNanum	koTenTen12
Portuguese	Freeling	ptTenTen11
Russian	RFTagger	ruTenTen11
Spanish	Freeling	esTenTen11

Table 1: Terminology support for languages in Sketch Engine in January 2014. POS tagger is mentioned as an important part of the corpus processing chain. The last column shows the corresponding default reference corpus.

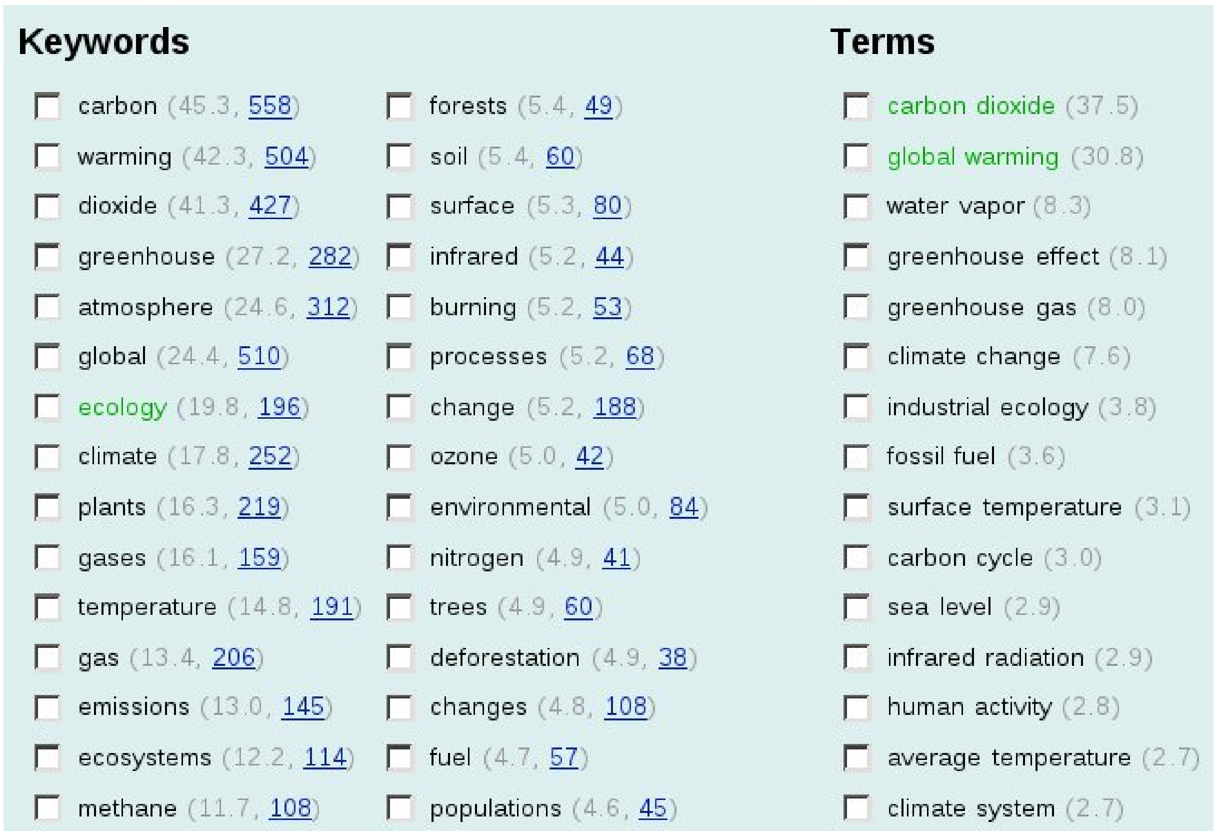
The display of term finding results is shown in Figure 1 for English, for a bootcatted climate-change corpus. Figure 2 shows a result set for Japanese in the mobile telecommunications domain, prepared for the first users of the systemm, the World Intellectual Property Organisation (WIPO), using their patents data, with their preferred display format.

The user can modify various extraction related options: Keyword reference corpus, term reference corpus, simple maths parameter, word length and other word properties, number of top results to display. The form is shown in Figure 3.

9 Current challenges

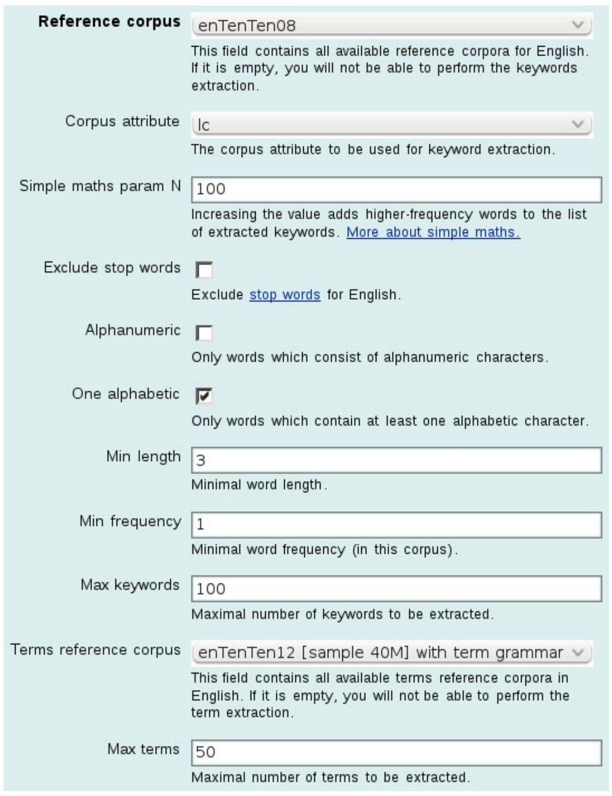
9.1 Canonical form: lemmas and word forms

In English one (almost) always wants to present each word in the term candidate in its canonical,



Keywords		Terms
carbon (45.3, 558)	forests (5.4, 49)	carbon dioxide (37.5)
warming (42.3, 504)	soil (5.4, 60)	global warming (30.8)
dioxide (41.3, 427)	surface (5.3, 80)	water vapor (8.3)
greenhouse (27.2, 282)	infrared (5.2, 44)	greenhouse effect (8.1)
atmosphere (24.6, 312)	burning (5.2, 53)	greenhouse gas (8.0)
global (24.4, 510)	processes (5.2, 68)	climate change (7.6)
ecology (19.8, 196)	change (5.2, 188)	industrial ecology (3.8)
climate (17.8, 252)	ozone (5.0, 42)	fossil fuel (3.6)
plants (16.3, 219)	environmental (5.0, 84)	surface temperature (3.1)
gases (16.1, 159)	nitrogen (4.9, 41)	carbon cycle (3.0)
temperature (14.8, 191)	trees (4.9, 60)	sea level (2.9)
gas (13.4, 206)	deforestation (4.9, 38)	infrared radiation (2.9)
emissions (13.0, 145)	changes (4.8, 108)	human activity (2.8)
ecosystems (12.2, 114)	fuel (4.7, 57)	average temperature (2.7)
methane (11.7, 108)	populations (4.6, 45)	climate system (2.7)

Figure 1: Term finding result in the Sketch Engine – keywords on the left, multiword terms on the right. The values in parentheses represent keyness score and frequency in the focus corpus. The green coloured candidates were used in a WebBootCaT run to build the corpus. The tickboxes are for specifying seed terms for iterating the corpus-building process.



Reference corpus	enTenTen08 This field contains all available reference corpora for English. If it is empty, you will not be able to perform the keywords extraction.
Corpus attribute	lc The corpus attribute to be used for keyword extraction.
Simple maths param N	100 Increasing the value adds higher-frequency words to the list of extracted keywords. More about simple maths.
Exclude stop words	☐ Exclude stop words for English.
Alphanumeric	☐ Only words which consist of alphanumeric characters.
One alphabetic	✓ Only words which contain at least one alphabetic character.
Min length	3 Minimal word length.
Min frequency	1 Minimal word frequency (in this corpus).
Max keywords	100 Maximal number of keywords to be extracted.
Terms reference corpus	enTenTen12 [sample 40M] with term grammar This field contains all available terms reference corpora in English. If it is empty, you will not be able to perform the term extraction.
Max terms	50 Maximal number of terms to be extracted.

Figure 3: Term finding settings form

dictionary form. But in French one does not. The top term candidate in one of our first experiments, using a French volcanoes corpus, was *nuée ardente*. The problem here is that *ardente* is the feminine form of the adjective, as required by the fact that *nuée* is a feminine noun. Simply taking the canonical form of each word (masculine singular, for adjectives) would flout the rule of adjective-noun gender agreement. A gender respecting lemma turns out necessary in such cases.

Noun lemmas beginning with a capital letter and gender respecting ending of adjectives had to be dealt with to correctly extract German phrases.

In most of the languages we have been working on, there are also some terms which should be given in the plural: an English example is *current affairs*. This is a familiar lexicographic puzzle: for some words, there are distinct meanings limited to some part or parts of the paradigm, and this needs noting. We are currently exploring options for this.

9.2 Versions of processing chains

If the version of the tools used for the reference corpus is not identical to the version used on the domain corpus, it is likely that the candidate list will be dominated by cases where the two versions treated the expression differently. Thus the two analyses of the expression will not match and (in simple cases), one of the analyses will have frequency zero in each corpus, giving one very high and one very low ratio. This makes the tool unusable if processing chains are not the same.

The reference corpus is processed in batch mode, and we hope not to upgrade it more than once a year. The domain corpus is processed at runtime. Until the development of the term-finding function, it did not greatly matter if different versions were used. For term-finding, we have had to look carefully at the tools, separating each out into an independent module, so that we can be sure of applying the same versions throughout. It has been a large task. (It also means that solutions based on POS-tagging by web services, where we do not control the web service, are not viable, since then, an unexpected upgrade to the web service will break our system.)

10 Evaluation

We have undertaken a first evaluation using the GENIA corpus (Kim et al., 2003), in which all terms have been manually identified.[5]

First, a plain-text version of GENIA was extracted and loaded into the system. Keyword and term extraction was performed to obtain the top 2000 keywords and top 1000 multi-word terms. Terms manually annotated in GENIA as well as terms extracted by our tool were normalized before comparison (lower case, spaces and hyphens removed) and then GENIA terms were looked up in the extraction results. 61 of the top 100 GENIA terms were found by the system. The terms not found were not English words: most were acronyms, e.g. EGR1, STAT-6.

Concerning the domain corpus size: Although the extraction method works well even with very small corpora (e.g. the sample environmental corpus in 1 consists of 100,000 words), larger corpora should be employed to cover more terms. An early version of this extraction tool was used to help lexicographers compile environment protection related terminology. A 50 million words corpus was sufficient in that case. (Avinesh et al., 2012) report 30 million words is enough.

[5]GENIA has also been used for evaluating term-finding systems by (Zhang et al., 2008).

11 Conclusion

We have built a system for finding terms in a domain corpus. It is currently set up for nine languages. In 2014 we shall extend the coverage of languages and improve the system according to further feedback from users.

Acknowledgement

This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013.

References

[Aker et al.2013] A. Aker, M. Paramita, and R. Gaizauskas. 2013. Extracting bilingual terminologies from comparable corpora. In *Proc. ACL*, pages 402–411.

[Avinesh et al.2012] PVS Avinesh, D. McCarthy, D. Glennon, and J. Pomikálek. 2012. Domain specific corpora from the web. In *Proc. EURALEX*.

[Baroni and Bernardini2004] M. Baroni and S. Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In *Proc. LREC*.

[Baroni et al.2006] M. Baroni, A. Kilgarriff, J. Pomikálek, and P. Rychlý. 2006. Webbootcat: instant domain-specific corpora to support human translators. In *Proc. EAMT*, pages 247–252.

[Gojun et al.2012] A. Gojun, U. Heid, B. Weissbach, C. Loth, and I. Mingers. 2012. Adapting and evaluating a generic term extraction tool. In *Proc. LREC*, pages 651–656.

[Jakubíček et al.2013] M. Jakubíček, A. Kilgarriff, V. Kovář, P. Rychlý, and V. Suchomel. 2013. The tenten corpus family. In *Proc. Corpus Linguistics*.

[Kilgarriff et al.2004] A. Kilgarriff, P. Rychlý, P. Smrž, and D. Tugwell. 2004. The sketch engine. *Proc. EURALEX*, pages 105–116.

[Kilgarriff2009] A. Kilgarriff. 2009. Simple maths for keywords. In *Proc. Corpus Linguistics*.

[Kim et al.2003] J-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

[Schulze and Christ1996] B. M. Schulze and O. Christ. 1996. The CQP user's manual. *Univ. Stuttgart*.

[Suchomel and Pomikálek2012] V. Suchomel and J. Pomikálek. 2012. Efficient web crawling for large text corpora. In *Proc. WAC7*, pages 39–43.

[Zhang et al.2008] Z. Zhang, J. Iria, C. A. Brewster, and F. Ciravegna. 2008. A comparative evaluation of term recognition algorithms. In *Proc. LREC*, pages 2108–2113.

A Graphical Interface for Automatic Error Mining in Corpora

Gregor Thiele Wolfgang Seeker Markus Gärtner Anders Björkelund Jonas Kuhn
Institute for Natural Language Processing
University of Stuttgart
{thielegr,seeker,gaertnms,anders,kuhn}@ims.uni-stuttgart.de

Abstract

We present an error mining tool that is designed to help human annotators to find errors and inconsistencies in their annotation. The output of the underlying algorithm is accessible via a graphical user interface, which provides two aggregate views: a list of potential errors in context and a distribution over labels. The user can always directly access the actual sentence containing the potential error, thus enabling annotators to quickly judge whether the found candidate is indeed incorrectly labeled.

1 Introduction

Manually annotated corpora and treebanks are the primary tools that we have for developing and evaluating models and theories for natural language processing. Given their importance for testing our hypotheses, it is imperative that they are of the best quality possible. However, manual annotation is tedious and error-prone, especially if many annotators are involved. It is therefore desirable to have automatic means for detecting errors and inconsistencies in the annotation.

Automatic methods for error detection in treebanks have been developed in the DECCA project[1] for several different annotation types, for example part-of-speech (Dickinson and Meurers, 2003a), constituency syntax (Dickinson and Meurers, 2003b), and dependency syntax (Boyd et al., 2008). These algorithms work on the assumption that two data points that appear in identical contexts should be labeled in the same way. While the data points in question, or *nuclei*, can be single tokens, spans of tokens, or edges between two tokens, context is usually modeled as n-grams over the surrounding tokens. A nucleus that occurs multiple times in identical contexts but is labeled differently shows variation and is considered a potential error.

Natural language is ambiguous and variation found by an algorithm may be a genuine ambiguity rather than an annotation error. Although we can support an annotator in finding inconsistencies in a treebank, these inconsistencies still need to be judged by humans. In this paper, we present a tool that allows a user to run automatic error detection on a corpus annotated with part-of-speech or dependency syntax.[2] The tool provides the user with a graphical interface to browse the variation nuclei found by the algorithm and inspect their label distribution. The user can always switch between high-level aggregate views and the actual sentences containing the potential error in order to decide if that particular annotation is incorrect or not. The interface thus brings together the output of the error detection algorithm with a direct access to the corpus data. This speeds up the process of tracking down inconsistencies and errors in the annotation considerably compared to working with the raw output of the original DECCA tools. Several options allow the user to fine-tune the behavior of the algorithm. The tool is part of ICARUS (Gärtner et al., 2013), a general search and exploration tool.[3]

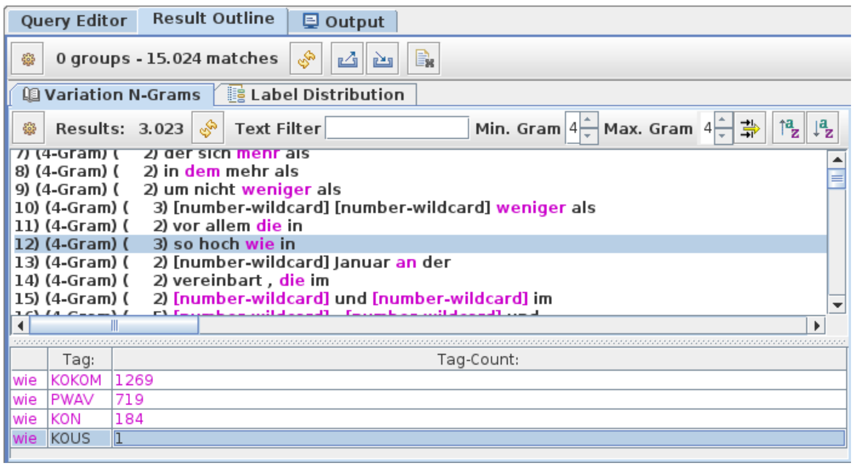
2 The Error Detection Algorithm

The algorithm, described in Dickinson and Meurers (2003a) for POS tags, works by starting from individual tokens (the nuclei) by recording their assigned part-of-speech over an entire treebank. From there, it iteratively increases the context for each instance by extending the string to both sides to include adjacent tokens. It thus successively builds larger n-grams by adding tokens to the left

[1]http://www.decca.osu.edu

[2]Generalizing the tool to support any kind of positional annotation is planned.

[3]http://www.ims.uni-stuttgart.de/data/icarus.html



```
7) (4-Gram) (    2) der sich mehr als
8) (4-Gram) (    2) in dem mehr als
9) (4-Gram) (    2) um nicht weniger als
10) (4-Gram) (    3) [number-wildcard] [number-wildcard] weniger als
11) (4-Gram) (    2) vor allem die in
12) (4-Gram) (    3) so hoch wie in
13) (4-Gram) (    2) [number-wildcard] Januar an der
14) (4-Gram) (    2) vereinbart , die im
15) (4-Gram) (    2) [number-wildcard] und [number-wildcard] im
```

	Tag:	Tag-Count:
wie	KOKOM	1269
wie	PWAV	719
wie	KON	184
wie	KOUS	1

Figure 1: The variation n-gram view.

or to the right. Instances are grouped together if their context is identical, i. e. if their token n-grams match. Groups where all instances have the same label do not show variation and are discarded. The algorithm stops when either no variation nuclei are left or when none of them can be further extended. All remaining groups that show variation are considered potential errors. Erroneous annotations that do not show variation in the data cannot be found by the algorithm. This limits the usefulness of the method for very small data sets. Also, given the inherent ambiguity of natural language, the algorithm is not guaranteed to exclusively output errors, but it achieves very high precision in experiments on several languages.

The algorithm has been extended to find errors in constituency and dependency structures (Dickinson and Meurers, 2003b; Boyd et al., 2008), where the definition of a nucleus is changed to capture phrases and dependency edges. Context is always modeled using n-grams over surrounding tokens, but see, e. g., Boyd et al. (2007) for extensions.

3 Graphical Error Mining

To start the error mining, a treebank and an error mining algorithm (part-of-speech or dependency) must be selected. The algorithm is then executed on the data to create the variation n-grams. The user can choose between two views for browsing the potential errors in the treebank: (1) a view showing the list of variation n-grams found by the error detection algorithm and (2) a view showing label distributions over word forms.

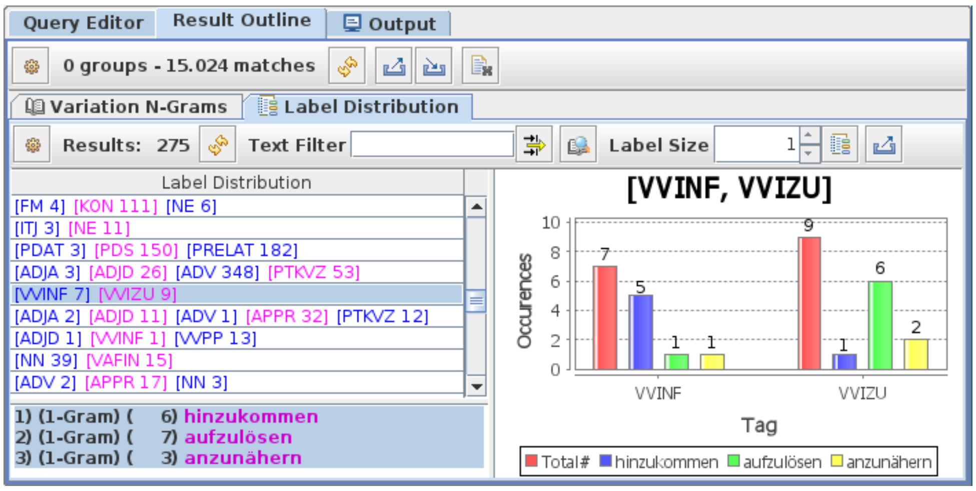
3.1 The Variation N-Gram View

Figure 1 shows a screenshot of the view where the user is presented with the list of variation n-grams output by the error detection algorithm. The main window shows the list of n-grams. When the user selects one of the n-grams, information about the nucleus is displayed below the main window. The user can inspect the distribution over labels (here part-of-speech tags) with their absolute frequencies. Above the main window, the user can adjust the length of the presented n-grams, sort them, or search for specific strings.

For example, Figure 1 shows a part of the variation n-grams found in the German TiGer corpus (Brants et al., 2002). The minimum and maximum length was restricted to four, thus the list contains only 4-grams. The 4-gram *so hoch wie in* was selected, which contains *wie* as its nucleus. In the lower part, the user can see that *wie* occurs with four different part-of-speech tags in the treebank, namely *KOKOM*, *PWAV*, *KON*, and *KOUS*. Note that the combination with *KOUS* occurs only once in the entire treebank.

Double clicking on the selected 4-gram in the list will open up a new tab that displays all sentences that contain this n-gram, with the nucleus being highlighted. The user can then go through each of the sentences and decide whether the annotated part-of-speech tag is correct. Each time the user clicks on an n-gram, a new tab will be created, so that the user can jump back to previous results without having to recreate them.

A double click on one of the lines in the lower part of the window will bring up all sentences that contain that particular combination of word form


Query Editor | Result Outline | Output

0 groups - 15.024 matches

Variation N-Grams | Label Distribution

Results: 275 Text Filter Label Size 1

Label Distribution

[FM 4] [KON 111] [NE 6]
[ITJ 3] [NE 11]
[PDAT 3] [PDS 150] [PRELAT 182]
[ADJA 3] [ADJD 26] [ADV 348] [PTKVZ 53]
[VVINF 7] [VVIZU 9]
[ADJA 2] [ADJD 11] [ADV 1] [APPR 32] [PTKVZ 12]
[ADJD 1] [VVINF 1] [VVPP 13]
[NN 39] [VAFIN 15]
[ADV 2] [APPR 17] [NN 3]

1) (1-Gram) (6) hinzukommen
2) (1-Gram) (7) aufzulösen
3) (1-Gram) (3) anzunähern

Figure 2: The label distribution view.

and part-of-speech tag. The fourth line will, for example, show the one sentence where *wie* has been tagged as *KOUS*, making it easy to quickly judge whether the tag is correct. In this case, the annotation is incorrect (it should have been *PWAV*) and should thus be marked for correction.

3.2 The Label Distribution View

In addition to the output of the algorithm by Dickinson and Meurers (2003a), the tool also provides a second view, which displays tag distributions of word forms to the user (see Figure 2). To the left, a list of unique label combinations is shown. Selecting one of them displays a list of word forms that occur with exactly these tags in the corpus. This list is shown below the list of label combinations. To the right, the frequencies of the different labels are shown in a bar chart. The leftmost bar for each label always shows the total frequency summed over all word forms in the set. Selecting one or more in the list of word forms adds additional bars to the chart that show the frequencies for each selected word form.

As an example, Figure 2 shows the tag combination *[VVINF][VVIZU]*, which are used to tag infinitives with and without incorporated *zu* in German. There are three word forms in the corpus that occur with these two part-of-speech tags: *hinzukommen*, *aufzulösen*, and *anzunähern*. The chart on the right shows the frequencies for each word form and part-of-speech tag, revealing that *hinzukommen* is mostly tagged as *VVINF* but once as *VVIZU*, whereas for the other two word forms it is the other way around. This example is interesting if one is looking for annotation errors in the TiGer treebank, because the two part-of-speech tags should have a complementary distribution (a German verb either incorporates *zu* or it does not).

Double clicking on the word forms in the list in the lower left corner will again open up a tab that shows all sentences containing this word form, regardless of their part-of-speech tag. The user may then inspect the sentences and decide whether the annotations are erroneous or not. If the user wants to see a specific combination, which is more useful if the total number of sentences is large, she can also click on one of the bars in the chart to get all sentences matching that combination. In the example, the one instance of *hinzukommen* being tagged as *VVIZU* is incorrect,[4] and the instances of the two other verbs tagged as *VVINF* are as well.

3.3 Dependency Annotation Errors

As mentioned before, the tool also allows the user to search for errors in dependency structures. The error mining algorithm for dependency structures (Boyd et al., 2008) is very similar to the one for part-of-speech tags, and so is the interface to the n-gram list or the distribution view. Dependency edges are therein displayed as triples: the head, the dependent, and the edge label with the edge's direction. As with the part-of-speech tags, the user can always jump directly to the sentences that contain a particular n-gram or dependency relation.

[4] Actually, the word form *hinzukommen* can belong to two different verbs, *hinzu-kommen* and *hin-kommen*. However, the latter, which incorporates *zu*, does not occur in TiGer.

4 Error Detection on TiGer

We ran the error mining algorithm for part-of-speech on the German TiGer Treebank (the dependency version by Seeker and Kuhn (2012)) and manually evaluated a small sample of n-grams in order to get an idea of how useful the output is.

We manually checked 115 out of the 207 variation 6-grams found by the tool, which amounts to 119 different nuclei. For 99.16% of these nuclei, we found erroneous annotations in the associated sentences. 95.6% of these are errors where we are able to decide what the right tag should be, the remaining ones are more difficult to disambiguate because the annotation guidelines do not cover them.

These results are in line with findings by Dickinson and Meurers (2003a) for the Penn Treebank. They show that even manually annotated corpora contain errors and an automatic error mining tool can be a big help in finding them. Furthermore, it can help annotators to improve their annotation guidelines by pointing out phenomena that are not covered by the guidelines, because these phenomena will be more likely to show variation.

5 Related Work

We are aware of only one other graphical tool that was developed to help with error detection in treebanks: Ambati et al. (2010) and Agarwal et al. (2012) describe a graphical tool that was used in the annotation of the Hindi Dependency Treebank. To find errors, it uses a statistical and a rule-based component. The statistical component is recall-oriented and learns a MaxEnt model, which is used to flag dependency edges as errors if their probability falls below a predefined threshold. In order to increase the precision, the output is post-processed by the rule-based component, which is tailored to the treebank's annotation guidelines. Errors are presented to the annotators in tables, also with the option to go to the sentences directly from there. Unlike the algorithm we implemented, this approach needs annotated training data for training the classifier and tuning the respective thresholds.

6 Conclusion

High-quality annotations for linguistic corpora are important for testing hypotheses in NLP and linguistic research. Automatically marking potential annotation errors and inconsistencies are one way of supporting annotators in their work. We presented a tool that provides a graphical interface for annotators to find and evaluate annotation errors in treebanks. It implements the error detection algorithms by Dickinson and Meurers (2003a) and Boyd et al. (2008). The user can view errors from two perspectives that aggregate error information found by the algorithm, and it is always easy to go directly to the actual sentences for manual inspection. The tool is currently extended such that annotators can make changes to the data directly in the interface when they find an error.

Acknowledgements

We thank Markus Dickinson for his comments. Funded by BMBF via project No. 01UG1120F, CLARIN-D, and by DFG via SFB 732, project D8.

References

Rahul Agarwal, Bharat Ram Ambati, and Anil Kumar Singh. 2012. A GUI to Detect and Correct Errors in Hindi Dependency Treebank. In *LREC 2012*, pages 1907–1911.

Bharat Ram Ambati, Mridul Gupta, Samar Husain, and Dipti Misra Sharma. 2010. A High Recall Error Identification Tool for Hindi Treebank Validation. In *LREC 2010*.

Adriane Boyd, Markus Dickinson, and Detmar Meurers. 2007. Increasing the Recall of Corpus Annotation Error Detection. In *TLT 2007*, pages 19–30.

Adriane Boyd, Markus Dickinson, and Detmar Meurers. 2008. On Detecting Errors in Dependency Treebanks. *Research on Language and Computation*, 6(2):113–137.

Sabine Brants, Stefanie Dipper, Silvia Hansen-Shirra, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *TLT 2002*, pages 24–41.

Markus Dickinson and W. Detmar Meurers. 2003a. Detecting Errors in Part-of-Speech Annotation. In *EACL 2003*, pages 107–114.

Markus Dickinson and W. Detmar Meurers. 2003b. Detecting Inconsistencies in Treebanks. In *TLT 2003*, pages 45–56.

Markus Gärtner, Gregor Thiele, Wolfgang Seeker, Anders Björkelund, and Jonas Kuhn. 2013. ICARUS – An Extensible Graphical Search Tool for Dependency Treebanks. In *ACL: System Demonstrations*, pages 55–60.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *LREC 2012*, pages 3132–3139.

DKIE: Open Source Information Extraction for Danish

Leon Derczynski
University of Sheffield
leon@dcs.shef.ac.uk

Camilla Vilhelmsen Field
University of Southern Denmark
cafie13@student.sdu.dk

Kenneth S. Bøgh
Aarhus University
ksb@cs.au.dk

Abstract

Danish is a major Scandinavian language spoken daily by around six million people. However, it lacks a unified, open set of NLP tools. This demonstration will introduce DKIE, an extensible open-source toolkit for processing Danish text. We implement an information extraction architecture for Danish within GATE, including integrated third-party tools. This implementation includes the creation of a substantial set of corpus annotations for data-intensive named entity recognition. The final application and dataset is made are openly available, and the part-of-speech tagger and NER model also operate independently or with the Stanford NLP toolkit.

1 Introduction

Danish is primarily spoken in the northern hemisphere: in Denmark, on the Faroe islands, and on Greenland. Having roots in Old Norse, Danish bears similarities to other Scandinavian languages, and shares features with English and German.

Previous tools and language resources for Danish have suffered from license restrictions, or from using small or non-reusable datasets. As a result, it is often difficult to use Danish language technologies, if anything is available at all. In cases where quality tools are available, they often have disparate APIs and input/output formats, making integration time-consuming and prone to error.

To remedy this, this paper presents an open-source information extraction toolkit for Danish, using the established and flexible GATE text processing platform (Cunningham et al., 2013). To this end, there are three main goals:

Adaptation: The application adapts to colloquial and formal Danish.

Interoperability: DKIE is internally consistent and adopts unified, well-grounded solutions to the problems of processing Danish. Where possible, DKIE re-uses existing components, and strives for compatibility with major text processing architectures.

Portability: It is preferable for developed components to be readily movable within the chosen architecture, GATE, and without, usable independently.

Openness: The resultant application, and corpora and annotations developed in its creation, are as freely-available as possible.

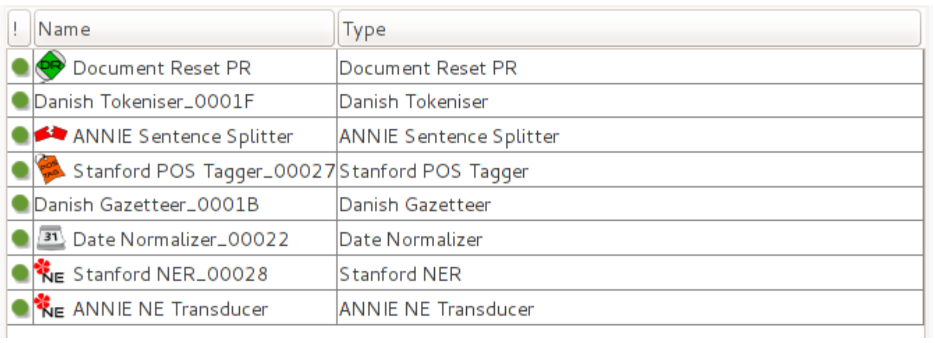
The remainder of this paper first discusses considerations specific to the language and prior work, then introduces the information extraction pipeline, followed by an evaluation of the tools provided.

2 Processing Danish

There are a few representational issues for Danish that are not solved in a unified fashion across existing technological issues. DKIE builds upon major standards in general linguistic annotation and in Danish to unify these solutions.

Danish is written using the Latin alphabet, with the addition of three vowels: æ, ø and å, which may be transliterated as ae, oe and aa respectively. It is similar to English in terms of capitalisation rules and character set.

Over time, the orthography of Danish has shifted. Among other things, a spelling reform in 1948 removed the capitalisation of nouns, and introduced the three vowel characters to represent existing vowel digraphs. There were also spelling shifts in this reform (e.g. *kjærlighed* to *kærlighed*). In addition, some towns and municipalities have changed the spelling of their name. For example, Denmarks second-largest city Aarhus changed its name to Århus with the 1948



!	Name	Type
●	Document Reset PR	Document Reset PR
●	Danish Tokeniser_0001F	Danish Tokeniser
●	ANNIE Sentence Splitter	ANNIE Sentence Splitter
●	Stanford POS Tagger_00027	Stanford POS Tagger
●	Danish Gazetteer_0001B	Danish Gazetteer
●	Date Normalizer_00022	Date Normalizer
●	Stanford NER_00028	Stanford NER
●	ANNIE NE Transducer	ANNIE NE Transducer

Figure 1: The ANNIE-based information extraction pipeline for Danish

reform, although Aalborg and Aabenraa did not. Later, in 2011, the city reverted from Århus to Aarhus. The city's university retained the Aarhus spelling throughout this period.

The effect of these relatively recent changes is that there exist digitised texts using a variety of orthographies not only to represent the same sound, as also in English, but also the same actual word. A language processing toolkit for Danish must exhibit sensitivity to these variances.

In addition, Danish has some word boundary considerations. Compound nouns are common (e.g. *kvindehåndboldlandsholdet* for "the women's national handball team"), as are hyphenated constructions (*fugle-fotografering* for "bird photography") which are often treated as single tokens.

Finally, abbreviations are common in Danish, and its acronyms can be difficult to disambiguate without the right context and language resource (e.g. OB for *Odense Boldklub*, a football club).

3 Background

The state of the art in Danish information extraction is not very interoperable or open compared to that for e.g. English. Previous work, while high-performance, is not available freely (Bick, 2004), or domain-restricted.[1] This makes results difficult to reproduce (Fokkens et al., 2013), and leads to sub-optimal interoperability (Lee et al., 2010). Even recent books focusing on the topic are heavily licensed and difficult for the average academic to access. Further, prior tools are often in the form of discrete components, hard to extend or to integrate with other systems.

Some good corpus resources are available, most recently the Copenhagen Dependency Treebank (CDT) (Buch-Kromann and Korzen, 2010), which built on and included previously-released corpora for Danish. This 200K-token corpus is taken from news articles and editorials, and includes document structure, tokenisation, lemma, part-of-speech and dependency relation information.

The application demonstrated, DKIE, draws only on open corpus resources for annotation, and the annotations over these corpora are released openly. Further, the application is also made open-source, with each component having similar or better performance when compared with the state-of-the-art.

4 Information Extraction Pipeline

This section details each step in the DKIE pipeline. A screenshot of the tool is shown in Figure 1.

4.1 Tokeniser

We adopt the PAROLE tokenisation scheme (Keson and Norling-Christensen, 1998). This makes different decisions from Penn Treebank in some cases, concatenating particular expressions as single tokens. For example, the two word phrase *i alt* – meaning *in total* – is converted to the single token `i_alt`. A set list of these group formations is given in the Danish PAROLE guidelines.

Another key difference is in the treatment of quoted phrases and hyphenation. Phrases connected in this way are often treated as single tokens. For example, the phrase *"Se og hør"-læserne* (*readers of "See and Hear"*, a magazine) is treated as a single token under this scheme.

4.2 Part-of-Speech tagger

We use a machine-learning based tagger (Toutanova et al., 2003) for Danish part-of-speech labelling. The original PAROLE

[1]E.g. CST's non-commercial-only anonymisation tool, at http://cst.dk/online/navnegenkender/

Tagger	Token accuracy %	Sentence acc. %
DKIE	95.3	49.1
TnT	96.2	39.1

Table 1: Part-of-speech labelling accuracy in DKIE

scheme introduces a set of around 120 tags, many of which are used only rarely. The scheme comprises tags built up of up to nine features. These features are used to describe information such as case, degree, gender, number, possessivity, reflexivity, mood, tense and so on (Keson and Norling-Christensen, 1998).

The PAROLE data includes morphological encoding in tags. We separate this data out in our corpus, adding morphological features distinct from part-of-speech data. This data may then be used by later work to train a morphological analyser, or by other tools that rely on morphological information.

We combine PAROLE annotations with the reduced tagset employed by the Danish Dependency Treebank (DDT) (Kromann, 2003). This has 25 tags. We adapted the tagger to Danish by including internal automatic mapping of æ, ø and å to two-letter diphthongs when both training and labelling, by adding extra sets of features for handling words and adjusting our unknown word threshold to compensate for the small corpus (as in Derczynski et al. (2013)), and by specifying the closed-class tags for this set and language. We also prefer a CRF-based classifier in order to get better whole-sequence accuracy, providing greater opportunities for later-stage tools such as dependency parsers to accurately process more of the corpus.

Results are given in Table 1, comparing token- and sentence-level accuracy to other work using the DDT and the TnT tagger (Brants, 2000). State-of-the-art performance is achieved, with whole-sentence tagging accuracy comparable to that of leading English taggers.

4.3 Gazetteers

High precision entity recognition can be achieved with gazetteer-based named entity recognition. This is a low-cost way of quickly getting decent performance out of existing toolkits. We include two special kinds of gazetteer for Danish. Firstly, it is important to annotation the names of entities specific to Denmark (e.g. Danish towns).

```
id      expression          interpretation
--      ----------          --------------
 3      igaa                ADD(DCT,day,-1)
13      Num._jul            ADD(DATE_MONTH_DAY(DCT, 12, 24),
                             day, TOKEN(0))
```

Figure 2: Example normalisation rules in TIMEN. "DCT" refers to the document creation time.

Secondly, entities outside of Denmark sometimes have different names specific to the Danish language (e.g. *Lissabon* for *Lisboa / Lisbon*).

As well as a standard strict-matching gazetteer, we include a "fuzzy" gazetteer specific to Danish that tolerates vowel orthography variation and the other changes introduced in the 1948 spelling reform. For locations, we extracted data for names of Danish towns from DBpedia and a local gazetteer, and from Wikipedia the Danish-language versions of the world's 1 000 most populous cities. For organisations, we used Wikipedia cross-language links to map the international organisations deemed notable in Wikipedia to their Danish translation and acroynm (e.g. the United Nations is referred to as *FN*). The major Danish political parties were also added to this gazetteer. For person names, we build lists of both notable people,[2] and also populated GATE's first and last name lists with common choices in Denmark.

4.4 Temporal Expression Annotation

We include temporal annotation for Danish in this pipeline, making DKIE the first temporal annotation tool for Danish. We follow the TimeML temporal annotation standard (Pustejovsky et al., 2004), completing just the TIMEX3 part.

Danish is interesting in that it permits flexible temporal anchors outside of reference time (Reichenbach, 1947) and the default structure of a calendar. For example, while in English one may use numbers to express a distance in days (*two days from now*) or into a month (*the second of March*), Danish permits these offsets from any agreed time. As a result, it is common to see expressions of the form *2. juledag*, which in this case is *the second christmas day* and refers to 26th December.

For this pipeline, we use finite state transducers to define how Danish timexes may be recognised. We then use the general-purpose TIMEN (Llorens et al., 2012) timex normalisation tool to provide calendar or TIMEX3 values for these expressions. Example rules are shown in Figure 2.

[2]See https://en.wikipedia.org/wiki/List_of_Danes, minus musicians due to stage names

4.5 Named entities

In addition to gazetteers, we present a machine learning-based approach to entity recognition and classification in Danish. We annotated the Copenhagen Dependency Treebank for person, location and organisation entities, according to the ACE guidelines (or as close as possible). This led to a total of 100 000 extra tokens annotated for NEs in Danish, doubling the previously-available amount. We used three annotators, achieving inter-annotator agreement of 0.89 on the first 100 000 tokens; annotation is an ongoing effort.

The data was used to learn a model tuned to Danish with an existing NER tool (Finkel et al., 2005). We removed word shape conjunctions features from the default configuration in an effort to reduced sensitivities introduced by the group noun tokenisation issue. This model, and the Stanford NER tool, were then wrapped as a GATE processing resource, contributing general-purpose Danish NER to the toolkit.

5 Conclusion

We will demonstrate a modern, interoperable, open-source NLP toolkit for information extraction in Danish. The released resources are: a GATE pipeline for Danish; tools for temporal expression recognition and normalisation for Danish; part-of-speech and named entity recognition models for Danish, that also work in the Stanford NLP architecture; and named entity corpus annotations over the Copenhagen Dependency Treebank.

Acknowledgments

This work was supported by EU funding under grant FP7-ICT-2013-10-611233, Pheme, and grant agreement No. 296322, AnnoMarket. We are grateful to Anders Søgaard of Copenhagen University for comments on an earlier draft and kind help with gazetteers. The first author would also like to thank Aarhus University for their kind provision of research facilities.

References

E. Bick. 2004. A named entity recognizer for Danish. In *Proceedings of LREC*.

T. Brants. 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. ACL.

M. Buch-Kromann and I. Korzen. 2010. The unified annotation of syntax and discourse in the Copenhagen Dependency Treebanks. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 127–131. ACL.

H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. 2013. Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. *PLoS computational biology*, 9(2):e1002854.

L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva. 2013. Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. In *Proceedings of Recent Advances in Natural Language Processing*. Association for Computational Linguistics.

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. ACL.

A. Fokkens, M. van Erp, M. Postma, T. Pedersen, P. Vossen, and N. Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1691–1701. Association for Computational Linguistics.

B. Keson and O. Norling-Christensen. 1998. PAROLE-DK. *The Danish Society for Language and Literature*.

M. T. Kromann. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, page 217.

K. Lee, L. Romary, et al. 2010. Towards interoperability of ISO standards for Language Resource Management. *Proc. ICGL 2010*.

H. Llorens, L. Derczynski, R. J. Gaizauskas, and E. Saquete. 2012. TIMEN: An Open Temporal Expression Normalisation Resource. In *LREC*, pages 3044–3051.

J. Pustejovsky, B. Ingria, R. Sauri, J. Castano, J. Littman, and R. Gaizauskas. 2004. The Specification Language TimeML. In *The Language of Time: A Reader*, pages 545–557. Oxford University Press.

H. Reichenbach. 1947. The tenses of verbs. In *Elements of Symbolic Logic*. Macmillan.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180. ACL.

Event Extraction for Balkan Languages

Vanni Zavarella, Dilek Küçük, Hristo Tanev
European Commission
Joint Research Centre
Via E. Fermi 2749
21027 Ispra (VA), Italy
first.last@jrc.ec.europa.eu

Ali Hürriyetoğlu
Center for Language Studies
Radboud University Nijmegen
P.O. Box 9103
NL-6500 HD Nijmegen
a.hurriyetoglu@let.ru.nl

Abstract

We describe a system for real-time detection of security and crisis events from online news in three Balkan languages: Turkish, Romanian and Bulgarian. The system classifies the events according to a fine-grained event type set. It extracts structured information from news reports, by using a blend of keyword matching and finite-state grammars for entity recognition. We apply a multilingual methodology for the development of the system's language resources, based on adaptation of language-independent grammars and on weakly-supervised learning of lexical resources. Detailed performance evaluation proves that the approach is effective in developing real-world semantic processing applications for relatively less-resourced languages.

1 Introduction

We describe a real-time event extraction system for three less-resourced languages: Bulgarian, Romanian and Turkish[1]. The goal of event extraction is to identify instances of a specified set of event types in natural language texts, and to retrieve database-like, structured information about event participants and attributes: these are the entities that are involved in the event and fill type-specific event roles (Ashish et al., 2006). For example, in the fragment *"Three workers were injured in a building collapse"*, the phrase *"three workers"* will be assigned a semantic role `Injured` of the event type `ManMadeDisaster` template.

Gathering and tracking such information over time from electronic news media plays a crucial role for the development of open-source intelligence systems, particularly in the context of global news monitoring of security threats, mass emergencies and disease outbreaks (Yangarber et al., 2005). In this view, it has been proved that being able to rely on highly multilingual text mining tools and language resources is of paramount importance, in order to achieve an unbiased coverage of global news content (Steinberger, 2012).

The system language components include finite state-based entity extraction grammars and domain-specific semantic lexica. These are adapted to the target language from existing language-independent resources or built by using semi-supervised machine learning algorithms, respectively. Most importantly, the lexical acquisition methods we put into place neither make use of any language knowledge nor require to have annotated corpora available.

Section 2 outlines the main processing stages of the application. In Section 3 we describe the methods applied to acquire and adapt the system's language knowledge bases. Finally, in Section 4 we report on an evaluation on event type classification and on the extraction of slot fillers for event templates, and we briefly discuss system performance and prospective improvements.

2 System Architecture

As depicted in Figure 1 (Tanev et al., 2009), first news feeds are clustered, upstream of the event extraction engine, by applying similarity metrics over meta data (named entities, locations, categories) extracted from single articles by dedicated, multilingual software.

Event extraction begins by preprocessing the title and first three sentences of each article within a cluster. This encompasses: fine-grained tokenization, sentence splitting, domain-specific dictionary look-up (i.e. matching of key terms indicating numbers, quantifiers, person titles, per-

[1]While belonging to three distant language families, namely Slavic, Romance and Turkic, respectively, they are spoken in the same geopolitical area, the Balkans.

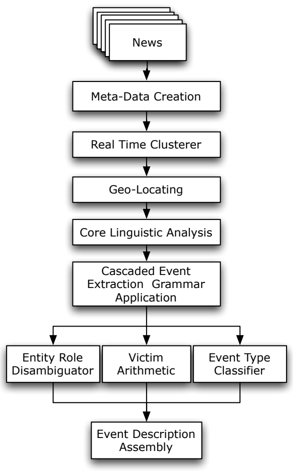


Figure 1: Event extraction processing chain

son groups descriptors like *civilians*, *policemen* and *Shiite*), and finally morphological analysis, simply consisting of lexicon look-up on large domain-independent morphological dictionaries from the MULTEXT project (Erjavec, 2004). Subsequently, a multi-layer cascade of finite-state extraction grammars in the ExPRESS formalism (Piskorski, 2007) is applied on such more abstract representation of the article text, in order to: a)identify entity referring phrases, such as persons, person groups, organizations, weapons, etc. b) assign them to event specific roles by linear combination with event triggering surface patterns. For example, in the text *"Iraqi policemen shot dead an alleged suicide bomber"* the grammar should extract the phrase *"Iraqi policemen"* and assign to it the semantic role `Perpetrator`, while the phrase *"alleged suicide bomber"* should be extracted as `Dead`. We use a "lexicon" of 1/2-slot patterns of the form:

```
<DEAD[Per]> was shot by <PERP>
<KIDNAP[Per]> has been taken hostage
```

where each slot position is assigned an event-specific semantic role and includes a type restriction (e.g. `Person`) on the entity which may fill the slot.

Finally, we aggregate and validate information extracted locally from each single article in the same cluster, such as entity role assignment, victim counts and event type.

We categorize the main event from each cluster with respect to a fine-grained event type set, shown in Table 1.

The event classification module consists of a blend of keyword matching, event role detection and a set of rules controlling their interaction. First, for each event type, we deploy: a) a list of weighted regular expression keyword patterns: each pattern match is awarded the corresponding weight, and an event type is triggered when the weight sum exceeds a defined threshold; b) a set of boolean pattern combinations: `OR` pattern lists are combined by the `AND` operator, each pattern is a restricted regular expression and conjunctions are restricted by proximity constraints. For example in order to detect *TerroristAttack* we use the following combination (translated here in English): ("bomb" `OR` "explosion" `OR`....) `AND` ("terrorist" `OR` "Al Qaida" `OR`..).

Besides the event `TYPE`, the other main slots of an output event frame include: `TYPE`, `DEAD`, `DEAD-COUNT`, `ARRESTED`, `ARRESTED-COUNT`, `PERPETRATOR`, `WEAPON`, etc.

The system will be demonstrated using a KML-aware earth browser[2]. Figure 2 shows a sample output event template.

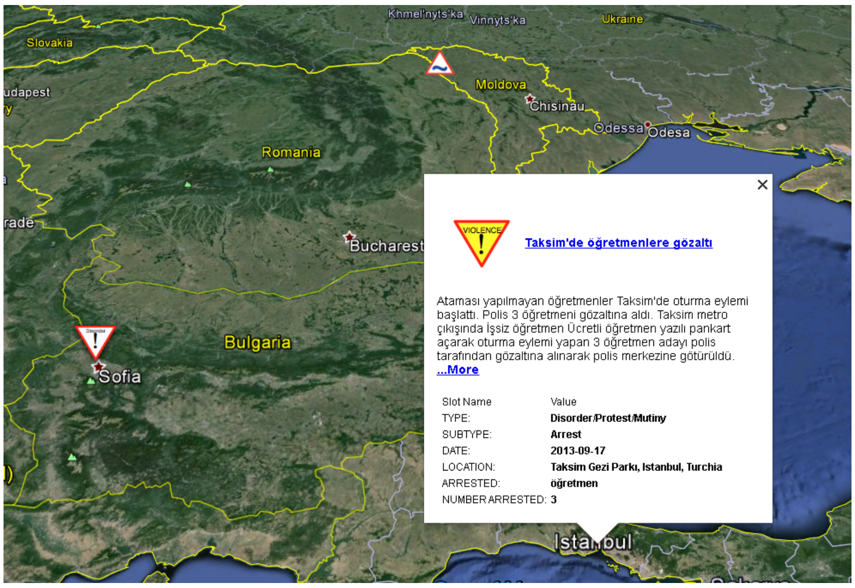
3 Development of language resources

The system's language components are:

Event grammar rules They consist of regular expressions over flat feature structures whose elements include, among the others, semantic types from the domain lexica. We use them to locally parse semantic entities such as person names, person group descriptions, and their clausal combination with verbal event patterns (see Section 2). Grammars in target languages are compiled by adapting the existing rules from source languages, such as English, while the bulk of grammar development mostly consists of providing suitable lexical resources.

Semantic dictionaries Domain-specific lexica, listing a number of (possibly multi-word) expressions sub-categorized into semantic classes relevant for the event domain, with limited or no linguistic annotation, are used by entity recognition grammar rules. Such lexica were created using the weakly supervised terminology extraction algorithm LexiClass (Ontopopulis), described in (Tanev et al., 2009). In order to enforce syntactic

[2]E.g. Google Earth. Notice that Geocoding is currently performed at the level of article text by a language-independent algorithm which is not yet integrated within the event detection process, while Document Creation Date is currently used as the event Date slot filler.



Taksim'de öğretmenlere gözaltı

Ataması yapılmayan öğretmenler Taksim'de oturma eylemi başlattı. Polis 3 öğretmeni gözaltına aldı. Taksim metro çıkışında İşsiz öğretmen Ücretli öğretmen yazılı pankart açarak oturma eylemi yapan 3 öğretmen adayı polis tarafından gözaltına alınarak polis merkezine götürüldü. ...More

Slot Name	Value
TYPE:	**Disorder/Protest/Mutiny**
SUBTYPE:	**Arrest**
DATE:	**2013-09-17**
LOCATION:	**Taksim Gezi Parkı, Istanbul, Turchia**
ARRESTED:	**öğretmen**
NUMBER ARRESTED:	**3**

Figure 2: A sample output template of the system

constraints (e.g. Case) into event clause rules for Romanian language, we have enriched learnt lexical entries for the semantic classes with morphological annotations, using MULTEXT resources. For Turkish, as we do not currently perform morphological analysis, we have rather included common inflected forms of the applicable lexical entries, resulting in larger lexica.

Event triggering patterns They are also acquired semi-automatically, starting with a set of seed examples and an article clustering, by deploying the paraphrase learning algorithm described in (Tanev et al., 2008). For Bulgarian, the grammar, semantic dictionaries and event patterns were created simultaneously, following a semi-automatic approach, described in (Tanev and Steinberger, 2013). In particular, we learned a list of terms referring to people, institutions and organizations and the corresponding pre- and post-modifiers (about 5000 terms). In the same manner, we learned about 550 surface patterns for killing, injuring, kidnapping and arresting actions, together with a 4 level grammar cascade.

Keyword terms The keyword sets used in the event type definitions, namely the OR lists in the boolean pattern combinations (see Section 2 above), can be viewed as instances of some more abstract semantic classes, that a domain expert uses to model a target event scenario. These classes are semi-automatically acquired using the LexiClass algorithm, and then manually combined. As Turkish is an agglutinative language, we have frequently added wildcards at the ends of keywords to cover possible inflected forms.

Table 1: Event type set

AirMissileAttack	Landslide
ArmedConflict	LightningStrike
Arrest	ManMadeDisaster
Assassination	MaritimeAccident
Avalanche	PhysicalAttack
BioChemicalAttack	Robbery
Bombing	Shooting
Disorder/Protest/Mutiny	Stabbing
Earthquake	Storm
Execution	TerroristAttack
Explosion	TropicalStorm
Floods	Tsunami
HeatWave	Vandalism
HeavyWeaponsFire	VolcanicEruption
HostageVideoRelease	Wildfire
HumanitarianCrisis	WinterStorm
Kidnapping	NONE

4 Experiments and Evaluation

System performance is evaluated on three different extractive tasks, carried out on the titles and first three sentences of single news articles: event type classification, event role name/description extraction and victim counting.

We collected test corpora of 52, 126 and 115 news articles for Bulgarian, Romanian and Turkish, respectively, spanning over a time range of 2 months[3]. For each article in the gold standard, we

[3]Articles were manually selected using news aggregators such as Google News. Type distribution resulted in zeroes for

Table 2: System performance in single article extraction mode.

Lang	Type	Dead			Injured			Arrested			Kidnapped			Perpetrator		Weapon	
	MRR	mF	MF	MSE	mF	MF	MSE	mF	MF	MSE	mF	MF	MSE	mF	MF	mF	MF
BG	0.34	0.27	0.68	17.08	0.44	0.6	108.82	0.22	1.0	7.69	0.4	0.5	0.71	0.0	0.0	0.39	1.0
RO	0.22	0.48	0.73	36.53	0.46	0.97	18.57	0.39	0.82	80.5	0.2	1.0	2.14	0.07	0.67	0.1	0.2
TR	0.66	0.73	0.79	16.41	0.85	0.91	0.24	0.31	0.36	52.17	0.4	0.33	0.82	0.25	0.67	0.77	1.0

annotated: a list of applicable types, ordered by relevance, for the main event reported in the article; the set of all the names/descriptions occurring in the text for each applicable event role, merging morphological variants; the cumulative count for the roles Dead, Injured, Kidnapped and Arrested.

Event type classification is evaluated by applying an adapted version of the mean reciprocal rank (MRR) score, used in Information Retrieval to evaluate processes producing a list of relevance ordered query responses. In our case, the MRR for a set of N articles is:

$$MRR = \frac{1}{|N|}\sum_{i=1}^{N}\frac{1}{rank_i}$$

where $rank$ is the rank of the system type response within the gold standard type list for each article.

For each role name/description extraction separately, we compute standard Precision, Recall and F1-measure on system responses, based on partial, n-gram match with gold standard responses, ignoring morphological suffixes.

Finally, we record the root Mean Squared Error (MSE) of system output victim count values against gold standard, over all applicable roles.

Table 2 summarizes the evaluation results. mF and MF columns for each role description task represent respectively the micro and macro average F1-measure over the test set.

Overall, the performance figures are in line with previous evaluations on other languages (Tanev et al., 2009). This proves the methodology is effective on adapting the system to new languages even with little lexical and syntactical proximity. Turkish system consistently outperforms the others, and it also underwent the most resource development cycles: this suggests that applying learning iterations, alternated with human filtering, to the language resources, can increase system accuracy, eventually making it usable for real-world applications. System accuracy is still unreliable for victim counting. One of the main reasons for large errors in victim counting is that the system interprets historical victim statistics reported in articles as event instances. We are currently implementing temporal and discourse heuristics to mitigate this problem.

some less frequent event types.

Acknowledgments

This study is supported in part by a postdoctoral research grant from TÜBİTAK and by the Dutch national program COMMIT.

References

Naveen Ashish, Doug Appelt, Dayne Freitag, and Dmitry Zelenko. 2006. Proceedings of the workshop on event extraction and synthesis. Technical report, AAAI.

Tomaz Erjavec. 2004. MULTEXT-East morphosyntactic specifications.

Jakub Piskorski. 2007. ExPRESS–extraction pattern recognition engine and specification suite. In *Proceedings of the International Workshop Finite-State Methods and Natural language Processing*.

Ralf Steinberger. 2012. A survey of methods to ease the development of highly multilingual text mining applications. *Language Resources and Evaluation*, 46(2):155–176.

Hristo Tanev and Josef Steinberger. 2013. Semi-automatic acquisition of lexical resources and grammars for event extraction in Bulgarian and Czech. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pages 110–118.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In E. Kapetanios, V. Sugumaran, and M. Spiliopoulou, editors, *Natural Language and Information Systems*, volume 5039 of *Lecture Notes in Computer Science*, pages 207–218.

Hristo Tanev, Vanni Zavarella, Jens Linge, Mijail Kabadjov, Jakub Piskorski, Martin Atkinson, and Ralf Steinberger. 2009. Exploiting Machine Learning Techniques to Build an Event Extraction System for Portuguese and Spanish. *Linguamática*, 1(2):55–66.

Roman Yangarber, Lauri Jokipii, Antti Rauramo, and Silja Huttunen. 2005. Extracting information about outbreaks of infectious epidemics. In *Proceedings of the HLT/EMNLP*.

Anaphora – Clause Annotation and Alignment Tool

Borislav Rizov
Department of Computational Linguistics, IBL-BAS
52 Shipchenski Prohod Blvd., bl. 17
1113 Sofia, Bulgaria
boby@dcl.bas.bg

Rositsa Dekova
Department of Computational Linguistics, IBL-BAS
52 Shipchenski Prohod Blvd., bl. 17
1113 Sofia, Bulgaria
rosdek@dcl.bas.bg

Abstract

The paper presents Anaphora – an OS and language independent tool for clause annotation and alignment, developed at the Department of Computational Linguistics, Institute for Bulgarian Language, Bulgarian Academy of Sciences. The tool supports automated sentence splitting and alignment and modes for manual monolingual annotation and multilingual alignment of sentences and clauses. Anaphora has been successfully applied for the annotation and the alignment of the Bulgarian-English Sentence- and Clause-Aligned Corpus (Koeva et al. 2012a) and a number of other languages including French and Spanish.

1 Introduction

For years now corpus annotation has played an essential part in the development of various NLP technologies. Most of the language resources, however, do not include clause annotation and alignment which are considered quite useful in recent research on Machine Translation (MT) and parallel text processing (Piperidis et al., 2000; Sudoh et al., 2010; Ramanathan et al., 2011).

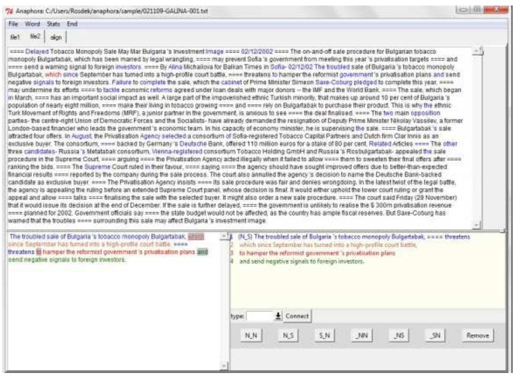
Aiming to facilitate and improve the process of clause annotation and alignment of multilingual texts, we developed Anaphora.

The tool is OS and language independent and supports automated sentence splitting and alignment, manual sentence and clause splitting, validation, correction and alignment, selection and annotation of conjunctions (including compounds (MWE)), and identification of the type of relation between pairs of syntactically connected clauses.

2 User Interface and Functionalities

Anaphora supports two kinds of operating modes: a monolingual and a multilingual one.

The monolingual mode is designed for manual editing and annotation of each part of the parallel corpus. The window consists of three active panes (Fig. 1): **Text view** (top pane), **Sentence view** (bottom left-hand pane) and **Clause view and annotation** (bottom right-hand pane).



==== Delayed Tobacco Monopoly Sale May Mar Bulgaria 's Investment Image ==== 02/12/2002 ==== The on-and-off sale procedure for Bulgarian tobacco monopoly Bulgartabak, which has been marred by legal wrangling, ==== may prevent Sofia 's government from meeting this year 's privatisation targets ==== and ==== send a warning signal to foreign investors. ==== By Alina Michailova for Balkan Times in Sofia- 02/12/02 The troubled sale of Bulgaria 's tobacco monopoly Bulgartabak, which since September has turned into a high-profile court battle, ==== threatens to hamper the reformist government 's privatisation plans and send negative signals to foreign investors. Failure to complete the sale, which the cabinet of Prime Minister Simeon Saxe-Coburg pledged to complete this year, ==== may undermine its efforts ==== to tackle economic reforms agreed under loan deals with major donors -- the IMF and the World Bank. ==== The sale, which began in March, ==== has an important social impact as well. A large part of the impoverished ethnic Turkish minority, that makes up around 10 per cent of Bulgaria 's population of nearly eight million, ==== make their living in tobacco growing ==== and ==== rely on Bulgartabak to purchase their product. This is why the ethnic Turk Movement of Rights and Freedoms (MRF), a junior partner in the government, is anxious to see ==== the deal finalised. ==== The two main opposition parties- the centre-right Union of Democratic Forces and the Socialists- have already demanded the resignation of Deputy Prime Minister Nikolay Vassilev, a former London-based financier who leads the government 's economic team. In his capacity of economy minister, he is supervising the sale. ==== Bulgartabak 's sale attracted four offers. In August, the Privatisation Agency selected a consortium of Sofia-registered Tobacco Capital Partners and Dutch firm Clar Innis as an exclusive buyer. The consortium, ==== backed by Germany 's Deutsche Bank, offered 110 million euros for a stake of 80 per cent. Related Articles ==== The other three candidates- Russia 's Metatabak consortium, Vienna-registered consortium Tobacco Holding GmbH and Russia 's Rosbulgartabak- appealed the sale procedure in the Supreme Court, ==== arguing ==== the Privatisation Agency acted illegally when it failed to allow ==== them to sweeten their final offers after ==== ranking the bids. ==== The Supreme Court ruled in their favour, ==== saying ==== the agency should have sought improved offers due to better-than-expected financial results ==== reported by the company during the sale process. The court also annulled the agency 's decision to name the Deutsche Bank-backed candidate as exclusive buyer. ==== The Privatisation Agency insists ==== its sale procedure was fair and denies wrongdoing. In the latest twist of the legal battle, the agency is appealing the ruling before an extended Supreme Court panel, whose decision is final. It would either uphold the lower court ruling or grant the appeal and allow ==== talks ==== finalising the sale with the selected buyer. It might also order a new sale procedure. ==== The court said Friday (29 November) that it would issue its decision at the end of December. If the sale is further delayed, ==== the government is unlikely to realise the $ 300m privatisation revenue ==== planned for 2002. Government officials say ==== the state budget would not be affected, as the country has ample fiscal reserves. But Saxe-Coburg has warned that the troubles ==== surrounding this sale may affect Bulgaria 's investment image.

The troubled sale of Bulgaria 's tobacco monopoly Bulgartabak, which since September has turned into a high-profile court battle, ==== threatens to hamper the reformist government 's privatisation plans and send negative signals to foreign investors.

1 (N_S) The troubled sale of Bulgaria 's tobacco monopoly Bulgartabak, ==== threatens
2 which since September has turned into a high-profile court battle,
3 to hamper the reformist government 's privatisation plans
4 and send negative signals to foreign investors.

Figure 1. Anaphora – monolingual mode.

In this mode the user may chose a file for verification and post-editing of the automatically performed sentence alignment. The individual monolingual texts which are part of an aligned pair are selected by the **file** tabs.

The monolingual mode offers the following functionalities:

- sentence splitting;
- clause splitting;
- correction of wrong splitting (merging of split sentences/clauses);
- annotation of conjunctions;
- selection of compounds;
- identification of the type of relation between pairs of syntactically connected clauses.

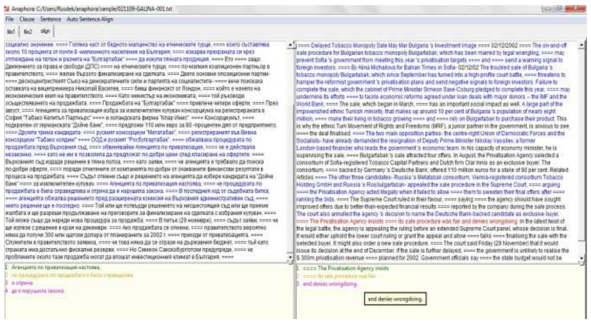
The end of a sentence may be changed by choosing the last word of the sentence and marking it using the End button from the top menu. Thus, the selection of the word as a sentence end is toggled and if it was marked as an End word, it is no longer such and the following sentence is automatically merged to the current one. If the word has not been already marked as an end, it is thus marked as one and the sentence is automatically split.

Clicking on any word of a sentence in the Text view pane results in the sentence appearing in the Sentence view pane, where clause segmentation and choice of conjunction are performed. The user defines the boundaries of clauses by selecting the words in them. This is achieved by marking the particular fragment of the text in the Sentence view pane with the mouse and pressing the 'space' key. This operation toggles the selection. Thus, a repeated use causes deselection. Marking a disconnected clause is done by marking the block of text containing it and unmarking the unnecessary words. When a clause is defined, it is listed in the bottom right-hand pane in a new color following the surface order of the sentence. Selection of a clause within another clause is also possible. Then the inner clause is listed directly after the split clause while the order of the split clause in the Clause view pane depends on the position of its first word in the sentence.

Once the clauses are defined, the user may annotate the conjunction of two clauses, also referred to as a marker. The marker may consist of one or more words or an empty word. Empty words (w="====") are artificial elements automatically introduced at the beginning of a potential new clause. An empty word may be selected as a marker when the conjunction is not explicit or the clauses are connected by means of a punctuation mark (for simplicity of annotation punctuation marks are not identified as independent tokens but are attached to the preceding token). When a word or a compound from one clause is selected in the Sentence view pane the user chooses another clause from the Clause view pane to create a pair of syntactically linked clauses. Then the relation for the pair is identified by selecting its type with the grey buttons N_N (coordination), N_S (subordinated clause following the main clause), S_N (subordinated clause preceding the main clause), etc.

The multilingual mode is selected with the **align** tab. In this mode annotators can create, validate and correct the alignment of the parallel units – sentences and/or clauses.

The window (Fig. 2) has two parallel **Text view** panes (on the top) and two parallel **List view** panes (in the bottom). Depending on the chosen menu (Clause or Sentence) the bottom panes show lists of aligned clauses or sentences.



Anaphora C:/Users/Rosdek/anaphora/sample/021109-GALINA-001.txt
File Clause Sentence Auto Sentence Align
file1 file2 align
and denies wrongdoing.

[illegible]

==== Delayed Tobacco Monopoly Sale May Mar Bulgaria 's Investment Image ==== 02/12/2002 ==== The on-and-off sale procedure for Bulgarian tobacco monopoly Bulgartabak, which has been marred by legal wrangling, ==== may prevent Sofia 's government from meeting this year 's privatisation targets ==== and ==== send a warning signal to foreign investors. ==== By Alina Michailova for Balkan Times in Sofia- 02/12/02 The troubled sale of Bulgaria 's tobacco monopoly Bulgartabak which since September has turned into a high-profile court battle, ==== threatens to hamper the reformist government 's privatisation plans and send negative signals to foreign investors. Failure to complete the sale, which the cabinet of Prime Minister Simeon Saxe-Coburg pledged to complete this year, ==== may undermine its efforts ==== to tackle economic reforms agreed under loan deals with major donors – the IMF and the World Bank. ==== The sale, which began in March, ==== has an important social impact as well. A large part of the impoverished ethnic Turkish minority, that makes up around 10 per cent of Bulgaria 's population of nearly eight million, ==== make their living in tobacco growing ==== and ==== rely on Bulgartabak to purchase their product. This is why the ethnic Turk Movement of Rights and Freedoms (MRF), a junior partner in the government, is anxious to see ==== the deal finalised. ==== The two main opposition parties- the centre-right Union of Democratic Forces and the Socialists- have already demanded the resignation of Deputy Prime Minister Nikolay Vassilev, a former London-based financier who leads the government 's economic team. In his capacity of economy minister, he is supervising the sale. ==== Bulgartabak 's sale attracted four offers. In August, the Privatisation Agency selected a consortium of Sofia-registered Tobacco Capital Partners and Dutch firm Clar Innis as an exclusive buyer. The consortium, ==== backed by Germany 's Deutsche Bank, offered 110 million euros for a stake of 80 per cent. Related Articles ==== The other three candidates- Russia 's Metatabak consortium, Vienna-registered consortium Tobacco Holding GmbH and Russia 's Rosbulgartabak- appealed the sale procedure in the Supreme Court, ==== arguing ==== the Privatisation Agency acted illegally when it failed to allow ==== them to sweeten their final offers after ==== ranking the bids. ==== The Supreme Court ruled in their favour, ==== saying ==== the agency should have sought improved offers due to better-than-expected financial results ==== reported by the company during the sale process. The court also annulled the agency 's decision to name the Deutsche Bank-backed candidate as exclusive buyer. ==== The Privatisation Agency insists ==== its sale procedure was fair and denies wrongdoing. In the latest twist of the legal battle, the agency is appealing the ruling before an extended Supreme Court panel, whose decision is final. It would either uphold the lower court ruling or grant the appeal and allow ==== talks ==== finalising the sale with the selected buyer. It might also order a new sale procedure. ==== The court said Friday (29 November) that it would issue its decision at the end of December. If the sale is further delayed, ==== the government is unlikely to realise the $ 300m privatisation revenue ==== planned for 2002. Government officials say ==== the state budget would not be

1 Агенцията по приватизация настоява,
2 че процедурата по продажбата е била справедлива
3 и отрича
4 да е нарушила закона.

1 ==== The Privatisation Agency insists
2 ==== its sale procedure was fair
3 and denies wrongdoing.

Figure 2. Anaphora – multilingual mode.

The multilingual mode uses the output of the monolingual sentence and clause splitting and supports the following functionalities:

- automated sentence alignment;
- manual sentence alignment;
- manual clause alignment.

Automated sentence alignment is available as a menu command (Auto Sentence Align) in the multilingual mode.

To switch to manual sentence or clause alignment the corresponding menu commands are used – Sentence and Clause.

In the sentence menu the two bottom panes show lists of aligned sentences, each pair in a distinct color. The user may correct the alignment by choosing one or more sentences in each of the bottom panes and pressing the 'space' button to create a new alignment bead.

In the clause menu, when a sentence is selected in one of the two Text panes, its clauses are listed in the respective bottom pane. The corresponding aligned sentence appears in the parallel top pane with its clauses listed in the bottom. Alignment is performed when the user chooses one or more clauses from each of the bottom panes and then presses the 'space' button. Thus a new clause alignment bead is created.

3 Applications

Anaphora was successfully used for the annotation and the alignment of the Bulgarian-English Sentence- and Clause-Aligned Corpus (Koeva et al. 2012a) which was created as a training and evaluation data set for automatic clause alignment in the task of exploring the effect of clause reordering on the performance of SMT (Koeva et al., 2012b).

Since its development the tool is continuously used for annotation and clause alignment of different parts of the Bulgarian-X language Parallel Corpus (Koeva et al. 2012c) covering a number of languages including French and Spanish.

4 Implementation

Anaphora was designed as a self-sufficient module for annotation and clause alignment within the multi-functional platform Chooser (Koeva et al. 2008) which supports various NLP tasks that involve corpora annotation.

The tool is a stand-alone single user application implemented in Python and it uses the standard GUI library *tkinter* (the Tcl/Tk python binding) which makes it highly OS independent.

5 Data Processing and Representation

5.1 Input Data

The used format is a flat xml with root element *text.* The text is a list of word elements with several attributes like ‘w’ – wordform, ‘l’ – lemma, ‘u’ – annotator, ‘t’ – timestamp, ‘e’ – sentence end, etc.

Special attributes are responsible for marking the compounds (MWE) and clauses. The words that are members of a compound share a common value for the attribute ‘p’ (parent). Similarly, the words in a clause share a common value for clause – ‘cl’.

This format is compatible with the other modules of the Chooser platform. Thus, one file can be annotated with several different types of annotation like POS, semantic annotation, etc.

The system provides import scripts for two formats – plain text and the output of the Bulgarian Language Processing Chain (Koeva and Genov, 2011) – a TSV/CSV family format, where the text is tokenized and lemmatized.

Sentence splitting depends on the format of the input text. If it is a plain text, sentence splitting is based on the presence of end of sentence punctuation (full stop, exclamation mark, and question mark) followed by a capital letter. When the file is of the TSV/CSV family format sentence splitting is part of the Language Processing Chain.

5.2 Automated Sentence Alignment

The automated sentence alignment is performed using the Gale-Church aligning algorithm (Gale and Church, 1993).

6 Conclusions and Future Work

We believe that, based on its design and functionalities, Anaphora can be easily used and it will perform well for any given pair of languages, that is, it is to a great extent language independent. The system can also be applied as it is for phrase segmentation and word and phrase alignment. However, if we want to include simultaneous alignment of words, phrases, and clauses the system needs to be adopted.

We work on including additional functionalities to facilitate corpora annotation and parallel text processing such as anaphora annotation.

Our future intentions include also publishing it as an Open Source code so that it can serve the NLP community.

Acknowledgments

The present paper was prepared within the project *Integrating New Practices and Knowledge in Undergraduate and Graduate Courses in Computational Linguistics* (BG051PO001-3.3.06-0022) implemented with the financial support of the *Human Resources Development* Operational Programme 2007-2013 co-financed by the European Social Fund of the European Union. The authors take full responsibility for the content of the present paper and under no conditions can the conclusions made in it be considered an official position of the European Union or the Ministry of Education, Youth and Science of the Republic of Bulgaria.

References

William A. Gale and Kenneth W. Church. 1993. A Program for Aligning Sentences in Bilingual Corpora, *Computational Linguistics* 19(1): 75–102.

Svetla Koeva and Angel Genov. 2011. Bulgarian Language Processing Chain. In: *Proceeding to The Integration of multilingual resources and tools in Web applications Workshop in conjunction with GSCL 2011*, University of Hamburg.

Svetla Koeva, Borislav Rizov, Svetlozara Leseva. 2008. Chooser - A Multi-Task Annotation Tool. In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), Marrakech, ELRA* electronic publication, 728-734.

Svetla Koeva, Borislav Rizov, Ekaterina Tarpomanova, Tsvetana Dimitrova, Rositsa Dekova, Ivelina Stoyanova, Svetlozara Leseva, Hristina Kukova, Angel Genov. 2012a. Bulgarian-English Sentence- and Clause-Aligned Corpus. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACHR-2),* Lisboa: Colibri, 51-62.

Svetla Koeva, Borislav Rizov, Ekaterina Tarpomanova, Tsvetana Dimitrova, Rositsa Dekova, Ivelina Stoyanova, Svetlozara Leseva, Hristina Kukova, and Angel Genov. 2012b. Application of clause alignment for statistical machine translation. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), Korea, 2012.*

Svetla Koeva, Ivelina Stoyanova, Rositsa Dekova, Borislav Rizov, and Angel Genov. 2012c. Bulgarian X-language parallel corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12).* N. Calzolari et al. (Eds.) Istanbul: ELRA, 2480-2486.

Stelios Piperidis, Harris Papageorgiou, and Sotiris Boutsis. 2000. From sentences to words and clauses. In *J. Veronis, editor, Parallel Text Processing, Alignment and Use of Translation Corpora,* Kluwer Academic Publishers, 117–138.

Ananthakrishnan Ramanathan, Pushpak Bhattacharyya, Karthik Visweswariah, Kushal Ladha and Ankur Gandhe. 2011. Clause-based reordering constraints to improve statistical machine translation. In *Proceedings of the 5th International Joint Conference on NLP, Thailand, November 8-13, 2011*, 1351–1355.

Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Tsutomu Hirao, Masaaki Nagata. 2010. Divide and translate: improving long distance reordering in statistical machine translation. In *Proceedings of the Joint 5thWorkshop on SMT and Metrics MATR,* 418–427.

SPARSAR: An Expressive Poetry Reader

Rodolfo Delmonte & Anton Maria Prati
artment of Language Studies & Department of Computer Scie
Ca' Foscari University - 30123, Venezia, Italy
delmont@unive.it

Abstract

We present SPARSAR, a system for the automatic analysis of poetry(and text) style which makes use of NLP tools like tokenizers, sentence splitters, NER (Name Entity Recognition) tools, and taggers. In addition the system adds syntactic and semantic structural analysis and prosodic modeling. We do a dependency mapping to analyse the verbal complex and determine Discourse Structure. Another important component of the system is a phonological parser to account for OOVWs, in the process of grapheme to phoneme conversion of the poem. We also measure the prosody of the poem by associating mean durational values in msecs to each syllable from a database of syllable durations; to account for missing syllables we built a syllable parser with the aim to evaluate durational values for any possible syllable structure. A fundamental component for the production of emotions is the one that performs affective and sentiment analysis. This is done on a line by line basis. Lines associated to specific emotions are then marked to be pronounced with special care for the final module of the system, which is reponsible for the production of expressive reading by a TTS module, in our case the one made available by Apple on their computers. Expressive reading is allowed by the possibility to interact with the TTS.

1 Introduction

We present SPARSAR, a system for poetry (and text) style analysis by means of parameters derived from deep poem (and text) analysis. We use our system for deep text understanding called VENSES(XXX,2005) for that aim. SPARSAR(XXX,2013a) works on top of the output provided by VENSES and is organized in three main modules which can be used also to analyse similarities between couples of poems by the same or different poet and similarities between collections of poems by a couple of poets. In addition to what is usually needed to compute text level semantic and pragmatic features, poetry introduces a number of additional layers of meaning by means of metrical and rhyming devices. For these reasons more computation is required in order to assess and evaluate the level of complexity that a poem objectively contains. We use prosodic durational parameters from a database of English syllables we produced for a prosodic speech recognizer (XXX,1990). These parameters are used to evaluate objective presumed syllable and feet prosodic distribution at line level. The sum of all of these data is then used to create a parameterized version of the poem to be read by a TTS, with an appropriate expressivity. Expressive reading is generated by combining syntactic, semantic, lexical and prosodic information. It is a well-known fact that TTS systems are unable to produce utterances with appropriate prosody(van Santen et al.,2003)[1]. Besides the general problems related to TTS reading normal texts, when a poem is inputted to the TTS the result is worsened by the internal rules which compute stanza boundaries as sentence delimiters. So every time there are continuations or enjambements from one stanza to the next the TTS will not be able to see it, and will produce a long pause. The TTS is also blind to line boundaries. More importantly, the TTS reads every sentence with the same tone, thus contributing an unpleasant repeated overall boring sense which does not correspond to the contents read. This is why sentiment analysis can be of help, together with semantic processing at discourse level.
As regards affective or emotional reading, then, the prosody of current TTS systems is neutral, and generally uses flat intonation contours. Producing "expressive" prosody will require modifying rhythm, stress patterns and intonation as described in section 4(see Kao & Jurafsky,2012).

[1] as he puts it, "The wrong words are emphasized, phrase boundaries are not appropriately indicated, and there is no prosodic structure for longer stretches of speech. As a result, comprehension is difficult and the overall listening experience is disconcerting…" (ibid.,1657).

The paper is organized as follows: here below a subsection contains a short state of the art limited though to latest publications; section 2 shortly presents SPARSAR; section 3 is dedicated to Prosody, Rhyming and Metrical Structure; a short state of the art of expressive reading is presented in section 4, which is devoted to TextToSpeech and parameters induction from the analysis. Eventually we present an evaluation, a conclusion and work for the future.

2 PARSAR - Automatic Analysis of Poetic Structure and Rhythm with Syntax, Semantics and Phonology

SPARSAR[8] produces a deep analysis of each poem at different levels: it works at sentence level at first, than at line level and finally at stanza level. The structure of the system is organized as follows: at first syntactic, semantic and grammatical functions are evaluated. Then the poem is translated into a phonetic form preserving its visual structure and its subdivision into lines and stanzas. Phonetically translated words are associated to mean duration values taking into account position in the word and stress. Taking into account syntactic and semantic information, we then proceed to "demote" word stress of dependent or functional words. At the end of the analysis of the poem, the system can measure the following parameters: mean verse length in terms of msec. and in number of feet. The latter is derived by a line and stanza representation of metrical structure. More on this topic below.

Another important component of the analysis of rhythm is constituted by the algorithm that measures and evaluates rhyme schemes at stanza level and then the overall rhyming structure at poem level. As regards syntax, we build chunks and dependency structures. To complete our work, we introduce semantics at two levels. On the one hand, we isolate verbal complex in order to verify propositional properties, like presence of negation, computing factuality from a crosscheck with modality, aspectuality – that we derive from our lexica – and tense. We also classify referring espressions by distinguishing concrete from abstract nouns, identifying highly ambiguous from singleton concepts (from number of possible meanings from WordNet and other similar repositories). Eventually, we carry out a sentiment analysis of every poem, thus contributing a three-way classification: neutral, negative, positive that can be used as a powerful tool for expressive purposes.

3 Rhetoric Devices, Metrical and Prosodic Structure

The second module takes care of rhetorical devices, metrical structure and prosodic structure. This time the file is read on a line by line level by simply collecting strings in a sequence and splitting lines at each newline character. In a subsequent loop, whenever two newlines characters are met, a stanza is computed. In order to compute rhetorical and prosodic structure we need to transform each word into its phonetic counterpart, by accessing the transcriptions available in the CMU dictionary. The Carnegie Mellon Pronouncing Dictionary is freely available online and includes American English pronunciation[2]. We had available a syllable parser which was used to build the VESD database of English syllables (XXX, 1999a) (Venice English Syllable Database) to be used in the Prosodic Module of SLIM, a system for prosodic self-learning activities(XXX,2010), which we use whenever we have a failure of our pronunciation dictionary which covers some 170,000 entries.

Remaining problems to be solved are related to ambiguous homographs like "import" (verb) and "import" (noun) and are treated on the basis of their lexical category derived from previous tagging; and Out Of Vocabulary Words (OOVW). If a word is not found in the dictionary, we try different capitalizations, as well as breaking apart hyphenated words, and then we check with simple heuristics, differences in spelling determined by British vs. American pronunciation. Then we proceed by morphological decomposition, splitting at first the word from its prefix and if that still does not work, its derivational suffix. As a last resource, we use an orthographically based version of the same dictionary to try and match the longest possible string in coincidence with our OOVW. Some words we had to reconstruct are: wayfare, gangrened, krog, copperplate, splendor, filmy, seraphic, unstarred, shrive, slipstream, fossicking, unplotted, corpuscle, thither, wraiths, etc. In some cases, the problem that made the system fail was the syllable which was not available in our database of syllable durations, VESD[3]. This problem has been coped with

[2] It is available online at <http://www.speech.cs.cmu.edu/cgi-bin/cmudict/>.

[3] In VESD, syllables have been collected from WSJCAM, the Cambridge version of the continuous speech recognition corpus produced from the Wall Street Journal, distributed by the Linguistic Data Consortium (LDC). We worked on a subset of 4165 sentences, with 70,694 words which consti-

by launching the syllable parser and then computing durations from the component phonemes, or from the closest similar syllable available in the database. We only had to add 12 new syllables for a set of approximately 500 poems that we computed to test the system.

3.1 Computing Metrical Structure and Rhyming Scheme

Any poem can be characterized by its rhythm which is also revealing of the poet's peculiar style. In turn, the poem's rhythm is based mainly on two elements: meter, that is distribution of stressed and unstressed syllables in the verse, presence of rhyming and other poetic devices like alliteration, assonance, consonance, enjambements, etc. which contribute to poetic form at stanza level.

We follow Hayward (1991) to mark a poetic foot by a numerical sequence that is an alternation of 0/1: "0" for unstressed and "1" for stressed syllables. The sequence of these sings makes up the foot and depending on number of feet one can speak of iambic, trochaic, anapestic, dactylic, etc. poetic style. But then we deepen our analysis by considering stanzas as structural units in which rhyming plays an essential role. Secondly we implement a prosodic acoustic measure to get a precise definition of rhythm. Syllables are not just any combination of sounds, and their internal structure is fundamental to the nature of the poetic rhythm that will ensue. The use of duration has allowed our system to produce a model of a poetry reader that we implement by speech synthesis. To this aim we assume that syllable acoustic identity changes as a function of three parameters:

- internal structure in terms of onset and rhyme which is characterized by number consonants, consonant clusters, vowel or diphthong
- position in the word, whether beginning, end or middle
- primary stress, secondary stress or unstressed

4 TTS and Modeling Poetry Reading

The other important part of the work regards using the previous analyses to produce intelligible, correct, appropriate and possibly pleasant or catchy poetry reading by a TextToSpeech system. In fact, the intention was more ambitious and was producing an "expressive" reading of a poem in the sense also intended by work reported in Ovesdotter & Sprout(2005), Ovesdotter(2005), Scherer(2003). In Ovesdotter & Sprout(2005), the authors present work on fairy tales, intended to use positive vs negative classification of sentences to produce a better reading. To that aim they used a machine learning approach, based on the manual annotation of some 185 children stories[4]. They reported accuracy results around 63% and F-score around 70%, which they explain may be due to a very low interannotator agreement, and to the fact that the dataset was too small. In Ovesdotter(2005) the author presents work on the perception of emotion based again on fairy tales reading by human readers. The experiment had the goal of checking the validity of the association of acoustic parameters to emotion types. Global acoustic features included F0, intensity, speech rate in number of words, feet, syllables per minute, fluency, i.e. number of pauses or silences. The results show some contradictory data for ANGRY state, but fully compliant data for HAPPY[5]. These data must be regarded as tendencies and are confirmed by experiments reported also in Scherer(2003) and Schröder(2001). However, it must be underlined that all researchers confirm the importance of semantic content, that is the meaning as a means for transmitting affective states.

The TTS we are now referring to is the one freely available under Mac OSX in Apple's devices. In fact, the output of our system can be used to record .wav or .mpeg files that can then be played by any sound player program. The information made available by the system is sufficiently deep to allow for Mac TTS interactive program to adapt the text to be read and model it

tute half of the total number of words in the corpus amounting to 133,080. We ended up with 113,282 syllables and 287,734 phones. The final typology is made up of 44 phones, 4393 syllable types and 11,712 word types. From word-level and phoneme-level transcriptions we produced syllables automatically by means of a syllable parser. The result was then checked manually.

[4] Features used to learn to distinguish "emotional" from "neutral" sentences, include (ibid., 582): first sentence in the story; direct speech; thematic story type (animal tale, ordinary folk-tale, jokes and anecdotes); interrogative and exclamative punctuation marks; sentence length in words; ranges of story progress; percent of semantic words (JJ, N, V, RB); V count in sentence, excluding participles; positive and negative words; WordNet emotion words; interjections and affective words; content BOW: N,V,JJ,RB words by POS.

[5] In particular, "angry" was associated with "decreased F0" and "decreased speech rate", but also an increased "pausing". On the contrary, "happy" showed an "increased F0, intensity, pausing" but a "decreased speech rate". "Happy" is similar to "surprised", while "angry" is similar to "sad".

accurately. We used the internal commands which can modify sensibly the content of the text to be read. The voices now available are pleasant and highly intelligible. We produced a set of rules that take into account a number of essential variables and parameter to be introduced in the file to be read. Parameters that can be modified include: Duration as Speaking Rate; Intonation from first word marked to a Reset mark; Silence introduced as Durational value; Emphasis at word level increasing Pitch; Volume from first word marked to a Reset mark, increasing intensity. We discovered that Apple's TTS makes mistakes when reading some specific words, which we then had to input to the system in a phonetic format, using the TUNE modality.
The rules address the following information:

- the title
- the first and last line of the poem
- a word is one of the phonetically spelled out words
- a word is the last word of a sentence and is followed by an exclamation/interrogative mark
- a word is a syntactic head (either at constituency or dependency level)
- a word is a quantifier, or marks the beginning of a quantified expression
- a word is a SUBJect head
- a word marks the end of a line and is (not) followed by punctuation
- a word is the first word of a line and coincides with a new stanza and is preceded by punctuation
- a line is part of a sentence which is a frozen or a formulaic expression with specific pragmatic content specifically encoded
- a line is part of a sentence that introduces new Topic, a Change, Foreground Relevance as computed by semantics and discourse relations
- a line is part of a sentence and is dependent in Discourse Structure and its Move is Down or Same Level
- a discourse marker indicates the beginning of a subordinate clause

5 Evaluation, Conclusion and Future Work

We have done a manual evaluation by analysing a randomly chosen sample of 50 poems out of the 500 analysed by the system. The evaluation has been made by a secondary school teacher of English literature, expert in poetry[6]. We asked the teacher to verify the following four levels of analysis: 1. phonetic translation; 2. syllable division; 3. feet grouping; 4. metrical rhyming structure. Results show a percentage of error which is around 5% as a whole, in the four different levels of analysis. A first prototype has been presented in(XXX,2013a), and improvements have been done since then; but more work is needed to tune prosodic parameters for expressivity rendering both at intonational and rhythmic level. The most complex element to control seems to be variations at discourse structure which are responsible for continuation intonational patterns vs. beginning of a new contour.

Reference

XXX. 1999. "Prosodic Modeling for Syllable Structures from the VESD - Venice English Syllable Database", in *Atti 9° Convegno GFS-AIA*, Venezia, 161-168.

XXX. 2008. "Speech Synthesis for Language Tutoring Systems", in V.Melissa Holland & F.Pete Fisher(eds.), (2008), *The Path of Speech Technologies in Computer Assisted Language Learning*, Routledge - Taylor and Francis Group-, New York, 123-150.

XXX, 2010. "Prosodic tools for language learning", *International Journal of Speech Technology*. 12(4):161-184.

XXX, 2013a. SPARSAR: a System for Poetry Automatic Rhythm and Style AnalyzeR, SLATE 2013, Demonstration Track.

XXX. 2005. "VENSES – a Linguistically-Based System for Semantic Evaluation", in J. Quiñonero-Candela et al.(eds.), 2005. *Machine Learning Challenges*. LNCS, Springer, Berlin, 344-371.

M. Hayward. 1991. "A connectionist model of poetic meter". *Poetics*, 20, 303-317.

Justine Kao and Dan Jurafsky. 2012. "A Computational Analysis of Style, Affect, and Imagery in Contemporary Poetry". in *Proc. NAACL Workshop on Computational Linguistics for Literature*.

Cecilia Ovesdotter Alm, Richard Sproat, 2005. "Emotional sequencing and development in fairy tales", In *Proceedings of the First International Conference on Affective Computing and Intelligent Interaction, ACII '05*.

Cecilia Ovesdotter Alm, 2005. "Emotions from text: Machine learning for text-based emotion prediction", In *Proceedings of HLT/EMNLP*, 347-354.

Jan van Santen, Lois Black, Gilead Cohen, Alexander Kain, Esther Klabbers,Taniya Mishra, Jacques de Villiers, and Xiaochuan Niu. 2003. "Applications of Computer Generated Expressive Speech for Communication Disorders", in *Proc. Eurospeech*, Geneva, 1657-1660.

K. R. Scherer. 2003. "Vocal communication of emotions: a review of research paradigms", *Speech Communication*, 40(1-2):227-256.

[6] I here acknowledge the contribution of XXX and thank her for the effort.

Annotating by Proving using SemAnTE

Assaf Toledo[1] **Stavroula Alexandropoulou**[1] **Sophie Chesney**[2]

Robert Grimm[1] **Pepijn Kokke**[1] **Benno Kruit**[3]

Kyriaki Neophytou[1] **Antony Nguyen**[1] **Yoad Winter**[1]

[1] - Utrecht University [2] - University College London [3] - University of Amsterdam
{a.toledo,s.alexandropoulou,y.winter}@uu.nl
sophie.chesney.10@ucl.ac.uk, {pepijn.kokke,bennokr}@gmail.com
{r.m.grimm,k.neophytou,a.h.nguyen}@students.uu.nl

Abstract

We present SemAnTE, a platform for marking and substantiating a semantic annotation scheme of textual entailment according to a formal model. The platform introduces a novel approach to annotation by providing annotators immediate feedback whether the data they mark are substantiated: for positive entailment pairs, the system uses the annotations to search for a formal logical proof that validates the entailment relation; for negative pairs, the system verifies that a counter-model can be constructed. By integrating a web-based user-interface, a formal lexicon, a lambda-calculus engine and an off-the-shelf theorem prover, this platform facilitates the creation of annotated corpora of textual entailment. A corpus of several hundred annotated entailments is currently in preparation using the platform and will be available for the research community.

1 Introduction

The Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006) advance the development of systems that automatically determine whether an entailment relation obtains between a naturally occurring text *T* and a manually composed hypothesis *H*. The RTE corpus (Bar Haim et al., 2006; Giampiccolo et al., 2008), which is currently the only available resource of textual entailments, marks entailment candidates as positive/negative.[1] For example:

Example 1

- T: The book contains short stories by the famous Bulgarian writer, Nikolai Haitov.
- H: Nikolai Haitov is a writer.[2]
- Entailment: Positive

This categorization does not indicate the linguistic phenomena that underlie entailment or their contribution to inferential processes. In default of a gold standard identifying linguistic phenomena triggering inferences, entailment systems can be compared based on their performance, but the inferential processes they employ to recognize entailment are not directly accessible and consequently cannot be either evaluated or improved straightforwardly.

We address this problem by elucidating some of the central inferential processes underlying entailments in the RTE corpus, which we model formally within a standard semantic theory. This allows us not only to indicate linguistic phenomena that are involved in the recognition of entailment by speakers, but also to provide formal proofs that substantiate the annotations and explain how the

[1]Pairs of sentences in RTE 1-3 are categorized in two classes: *yes-* or *no-entailment*; pairs in RTE 4-5 are categorized in three classes: *entailment*, *contradiction* and *unknown*. We label the judgments *yes-entailment* from RTE 1-3 and *entailment* from RTE 4-5 as *positive*, and the other judgments as *negative*.

[2]Pair 622 from the development set of RTE 2.

modeled phenomena interact and contribute to the recognition process. In this sense the we adopt an *Annotating by Proving* approach to textual entailment annotation.

The annotation work is done using the SemAnTE (Semantic Annotation of Textual Entailment) platform, which incorporates a web-based user-interface, a formal lexicon, a lambda-calculus engine and an off-the-shelf theorem prover. We are currently using this platform to build a new corpus of several hundred annotated entailments comprising both positive and negative pairs. We decided to focus on the semantic phenomena of appositive, restrictive and intersective modification as these semantic phenomena are prevalent in the RTE datasets and can be annotated with high consistency, and as their various syntactic expressions can be captured by a limited set of concepts.[3] In the future, we plan to extend this sematic model to cover other, more complex phenomena.

2 Semantic Model

To model entailment in natural language, we assume that entailment describes a *preorder* on sentences. Thus, any sentence trivially entails itself (reflexivity); and given two entailments $T_1 \Rightarrow H_1$ and $T_2 \Rightarrow H_2$ where H_1 and T_2 are identical sentences, we assume $T_1 \Rightarrow H_2$ (transitivity). We use a standard model-theoretical extensional semantics, based on the simple *partial order* on the domain of *truth-values*. Each model M assigns sentences a truth-value in the set $\{0, 1\}$. Such a Tarskian theory of entailment is considered adequate if the intuitive entailment preorder on sentences can be described as the pairs of sentences T and H whose truth-values $[\![T]\!]^M$ and $[\![H]\!]^M$ satisfy $[\![T]\!]^M \leq [\![H]\!]^M$ for all models M.

We use annotations to link between textual representations in natural language and model-theoretic representations. This link is established by marking the words and structural configurations in T and H with lexical items that encode semantic meanings for the linguistic phenomena that we model. The lexical items are defined formally in a lexicon, as illustrated in Table 1 for major lexical categories over type:s e for *entities*, t for *truth-values*, and their functional compounds.

[3]This conclusion is based on an analysis of RTE 1-4, in which these modification phenomena were found to occur in 80.65% of the entailments and were annotated with cross-annotator agreement of 68% on average.

Category	Type	Example	Denotation
Proper Name	e	Dan	**dan**
Indef. Article	$(et)(et)$	a	A
Def. Article	$(et)e$	the	ι
Copula	$(et)(et)$	is	IS
Noun	et	book	**book**
Intrans. verb	et	sit	**sit**
Trans. verb	eet	contain	**contain**
Pred. Conj.	$(et)((et)(et))$	and	AND
Res. Adj.	$(et)(et)$	short	$R_m(\mathbf{short})$
Exist. Quant.	$(et)(et)t$	some	SOME

Table 1: Lexicon Illustration

Denotations that are assumed to be arbitrary are given in boldface. For example, the intransitive verb *sit* is assigned the type et, which describes functions from entities to a truth-values, and its denotation **sit** is an arbitrary function of this type. By contrast, other lexical items have their denotations restricted by the given model M. As illustrated in Figure 1, the coordinator *and* is assigned the type $(et)((et)(et))$ and its denotation is a function that takes a function A of type et and returns a function that takes a function B, also of type et, and returns a function that takes an entity x and returns 1 if and only if x satisfies both A and B.

$$\text{A} = \text{IS} = \lambda A_{et}.A$$
$$\iota = \lambda A_{et}.\begin{cases} a & A = (\lambda x_e.x = a) \\ \text{undefined} & \text{otherwise} \end{cases}$$
$$\text{WHO}_A = \lambda A_{et}.\lambda x_e.\iota(\lambda y.y = x \wedge A(x))$$
$$R_m = \lambda M_{(et)(et)}.\lambda A_{et}.\lambda x_e.M(A)(x) \wedge A(x)$$
$$\text{SOME} = \lambda A_{et}.\lambda B_{et}.\exists x.A(x) \wedge B(x)$$
$$\text{AND} = \lambda A_{et}.\lambda B_{et}.\lambda x_e.A(x) \wedge B(x)$$

Figure 1: Functions in the Lexicon

By marking words and syntactic constructions with lexical items, annotators indicate the underlying linguistic phenomena in the data. Furthermore, the formal foundation of this approach allows annotators to verify that the entailment relation (or lack thereof) that obtains between the textual forms of T and H also obtains between their respective semantic forms. This verification guarantees that the annotations are sufficient in the sense of providing enough information for recognizing the entailment relation based on the semantic abstraction. For example, consider the simple entailment *Dan sat and sang* $\Rightarrow$ *Dan sang* and assume annotations of *Dan* as a proper name, *sat* and *sang* as intransitive verbs and *and* as predicate conjunction. The formal model can be used to verify these annotations by constructing a proof as follows: for each model M:

$[\![\textit{Dan [sat [and sang]]}]\!]^M$

$= ((\text{AND}(\mathbf{sing}))(\mathbf{sit}))(\mathbf{dan})$	analysis
$= (((\lambda A_{et}.\lambda B_{et}.\lambda x_e.A(x) \wedge B(x))(\mathbf{sing}))(\mathbf{sit}))(\mathbf{dan})$	def. of AND
$= \mathbf{sit}(\mathbf{dan}) \wedge \mathbf{sing}(\mathbf{dan})$	func. app. to **sing**, **sit** and **dan**
$\leq \mathbf{sing}(\mathbf{dan})$	def. of $\wedge$
$= [\![\textit{Dan sang}]\!]^M$	analysis

3 Platform Architecture

The platform's architecture is based on a client-server model, as illustrated in Figure 2.

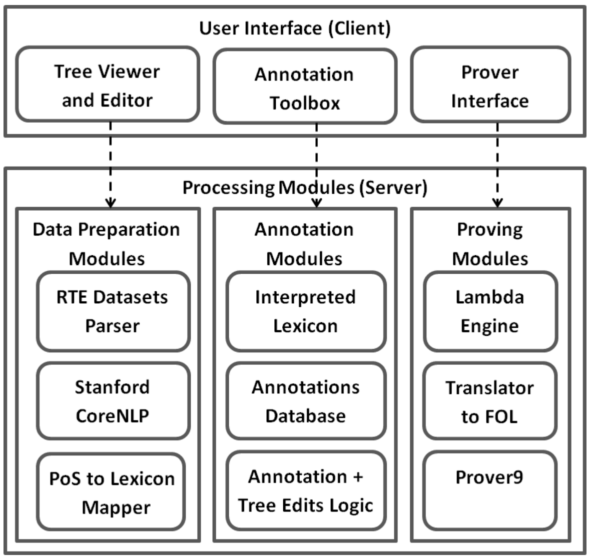


Figure 2: Platform Architecture

The user interface (UI) is implemented as a web-based client using Google Web Toolkit (Olson, 2007) and allows multiple annotators to access the RTE data, to annotate, and to substantiate their annotations. These operations are done by invoking corresponding remote procedure calls at the server side. Below we describe the system components as we go over the work-flow of annotating Example 1.

Data Preparation: We extract T-H pairs from the RTE datasets XML files and use the Stanford CoreNLP (Klein and Manning, 2003; Toutanova et al., 2003; de Marneffe et al., 2006) to parse each pair and to annotate it with part-of-speech tags.[4] Consequently, we apply a naive heuristic to map the PoS tags to the lexicon.[5] This process is called as part of the platform's installation and when annotators need to simplify the original RTE data in order to avoid syntactic/semantic phenomena that the semantic engine does not support. For example, the bare plural *short stories* is simplified to *some short stories* as otherwise the engine is unable to determine the quantification of this noun.

Annotation: The annotation is done by marking the tree-leaves with entries from the lexicon. For example, *short* is annotated as a restrictive modifier (*MR*) of the noun *stories*, and *contains* is annotated as a transitive verb (*V_2*). In addition, annotators manipulate the tree structure to fix parsing mistakes and to add leaves that mark semantic relations. For instance, a leaf that indicates the apposition between *the famous Bulgarian writer* and *Nikolai Haitov* is added and annotated as WHO_A. The server stores a list of all annotation actions. Figure 3 shows the tree-view, lexicon, prover and annotation history panels in the UI.

Proving: When annotating all leaves and manipulating the tree structures of *T* and *H* are done, the annotators use the prover interface to request a search for a proof that indicates that their annotations are substantiated. Firstly, the system uses lambda calculus reductions to create logical forms that represent the meanings of T and H in higher-order logic. At this stage, type errors may be reported due to erroneous parse-trees or annotations. In this case an annotator will fix the errors and re-run the proving step. Secondly, once all type errors are resolved, the higher-order representations are lowered to first order and Prover9 (McCune, 2010) is executed to search for a proof between the logical expressions of T and H.[6] The proofs are recorded in order to be included in the corpus release. Figure 4 shows the result of translating T and H to an input to Prover9.

4 Corpus Preparation

We have so far completed annotating 40 positive entailments based on data from RTE 1-4. The annotation is a work in progress, done by four Master students of Linguistics who are experts in the data and focus on entailments whose recognition relies on a mixture of appositive, restrictive or intersective modification. As we progress towards the compilation of a corpus of several hundred pairs, we extend the semantic model to support more inferences with less phenomena simplification.

[4]Version 1.3.4

[5]This heuristic is naive in the sense of not disambiguating verbs, adjectives and other types of terms according to their semantic features. It is meant to provide a starting point for the annotators to correct and fine-tune.

[6]Version 2009-11A

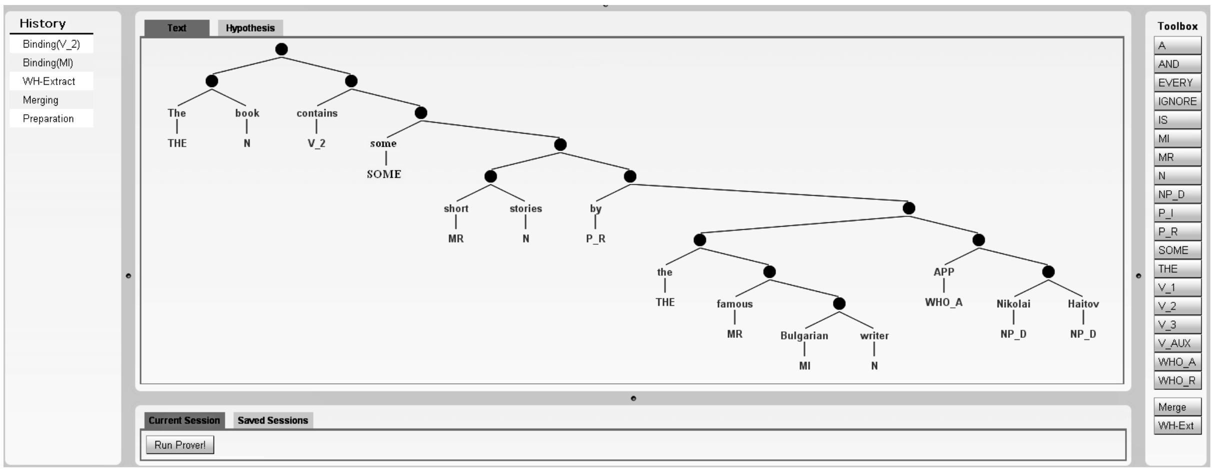


Figure 3: User Interface Panels: Annotation History, Tree-View, Prover Interface and Lexicon Toolbox

formulas(assumptions).
all x0 (((writer(x0) & bulgarian(x0)) & famous_writer_bulgarian(x0)) ↔ x0=c1).
all x0 (((stories(x0) & short_stories(x0)) & exists x1 (by_stories_short_stories(x1, x0) & (x1=c1 & x1=Nikolai_Haitov))) ↔ x0=c2). all x0 (book(x0) ↔ x0=c3).
contains(c2, c3).
end_of_list.

formulas(goals).
exists x0 (writer(x0) & x0=Nikolai_Haitov).
end_of_list.

Figure 4: Input for Theorem Prover

5 Conclusions

We introduced a new concept of an annotation platform which implements an *Annotating by Proving* approach. The platform is currently in use by annotators to indicate linguistic phenomena in entailment data and to provide logical proofs that substantiate their annotations. This method guarantees that the annotations constitute a complete description of the entailment relation and can serve as a gold-standard for entailment recognizers. The new corpus will be publicly available.

Acknowledgments

The work of Stavroula Alexandropoulou, Robert Grimm, Sophie Chesney, Pepijn Kokke, Benno Kruit, Kyriaki Neophytou, Antony Nguyen, Assaf Toledo and Yoad Winter was supported by a VICI grant number 277-80-002 by the Netherlands Organisation for Scientific Research (NWO).

References

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*. The Stanford Natural Language Processing Group.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Elena Cabrio. 2008. The fourth pascal recognising textual entailment challenge. In *TAC 2008 Proceedings*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. ACL.

William McCune. 2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Steven Douglas Olson. 2007. *Ajax on Java*. O'Reilly Media.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. ACL.

Answering List Questions using Web as a corpus

Patrícia Nunes Gonçalves, António Branco
University of Lisbon
Edifício C6, Departamento de Informática
Faculdade de Ciências, Universidade de Lisboa
Campo Grande, 1749-016 Lisboa, Portugal
{patricia.nunes, antonio.branco}@di.fc.ul.pt

Abstract

This paper supports the demo of LX-ListQuestion, a Web Question Answering System that exploit the redundancy of information available in the Web to answer List Questions in the form of Word Cloud.

1 Introduction

The combination of web growth and improvements in Information Technology has reignited the interest in Question Answering (QA) systems. QA is a type of information retrieval combined with natural language processing techniques that aims at finding exact answers to natural language questions. In a search engine, the user inserts a few keywords and gets as a result links and snippets. The task of finding the desired answer among the results that were returned then falls on the user. From the point of view of QA, in turn, the users use a question in natural language and the system searches within the documents for the answers.

Questions have various levels of complexity. When thinking about questions immediately come to mind factual questions (eg: When did Nelson Mandela die?), however, the QA area has expanded beyond factual questions towards more complex questions. One of the most common types of complex factual is list questions. The list questions are questions for which there is a list of answers, e.g., *In which countries Portuguese is an official language?* List answers: *Angola, Brazil, Cape Verde, Guine Bissau, Mozambique, Portugal, Macau, Sao Tome and Principe and East Timor.*

When the information that is needed is non-trivial and it is found spread over several texts, a lot of human effort is required to gather the various separate pieces of data into the desired result, which is not an easy task. Ideally, they would prefer to quickly get a precise answer and go on to make use of it instead of spending time searching and compiling the answer from pieces spread over several documents.

Our purpose is to provide better QA solutions to users, who desire direct answers to their queries, using approaches that deal with the complex problem of extracting answers found spread over several documents and use them to compile a list of answers that are the most accurate possible. The LX-ListQuestion development is guided by the circumstance that answers may appear redundantly in many places and in many forms. Our approach to address the problem of answering list questions is to explore this redundancy. To build on this redundancy, we use techniques that will be explained in section 4.

2 Related Work

List QA is an emerging topic and few approaches have been developed. The most common approach is to take a QA system for factoid questions and extend it to answer List questions. Some pioneering systems using this approach are (Gaizauskas et al., 2005) and (Wu and Strzalkowski, 2006), which show a low performance. Other systems explore NLP tools and linguistic resources (Hickl et al., 2006) (Yang et al., 2003). This approach seems to have achieved competitive results. However the time required for processing is very high and the performance of these systems depend on the performance of the supporting NLP tools.

Other approaches resort to statistical and machine learning approaches. The system developed in (Whittaker et al., 2006) is based on a statistical model. The system developed by (Yang and Chua, 2004) employ classification techniques to find complete and distinct answers. The system proposed by (Razmara and Kosseim, 2008) answers List questions using a clustering method to group candidate answers that co-occur more often in the collection.

Systems that take advantage from semantic content to answer List questions (Cardoso et al., 2009), (Hartrumpf and Leveling, 2010), (Dornescu, 2009) achieved good results although all information should be stored in the database. This approach seems suitable to QA system that focus on a specific domain where the information source can be limited and more easily stored.

3 List Question

In the context of QA research, list questions may appear in three basic forms: (1) a question starting with an interrogative pronoun, (2) a request using an imperative verb and (3) other forms: without interrogative pronoun or imperative verb (usually a complex noun phrase). Table 1 shows some examples.

Type of List Question	Example
Interrogative Pronoun	What European Union countries have national parks in the Alps?
Imperative Form	Name rare diseases with dedicated research centers in Europe.
Other	Chefs born in Austria who have received a Michelin Star.

Table 1: Examples of List questions

Another important topic in QA is to identify the so called question target. Our study shows that target can be expressed by a named entity, a common noun or be multiple targets. Most List Questions have named entities as target question, e.g. *Which cities in Germany have more than one university?*. Common noun is not so frequent as target question but it can show up, e.g. *Typical food of the Cape Verde cuisine.* Multiples target can be of named entities or common nouns, e.g. *Newspapers, magazines and other periodicals of Macau.*

The list answers (for a list question) may appear in many places and in many forms. They can be in the same document; when the answer is already a list, e.g., *list of cities in Portugal: Lisbon, Coimbra, Porto e Faro*; or the answers can be spread over multiple documents; e.g., (document A): *Lisbon is the capital of Portugal.* (document B) *Porto is a very important city in Portugal.* In the latter case, a QA system able to answer List questions has to deal with this diversity and find all answers of several texts and compose the final list of answers.

Our system focuses on answering List questions where the answers are extracted from several documents from the Web.

4 LX-ListQuestion Architecture

LX-ListQuestion System seeks to answer List questions through the use of the techniques of Question Answering running over the Web of Portuguese pages, while ensuring that the Final Answer List is as correct and complete as possible. This system has three main modules: Question Processing, Passage Retrieval and Answer Extraction. Figure 1 shows its architecture.

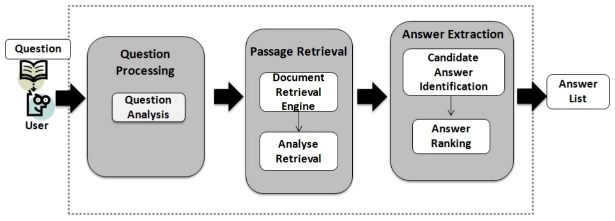


Figure 1: Question Answering System Architecture

The Question Processing module is responsible for converting a natural language question into a form that a computer is capable of handling and extracting the information that will be passed and used by the subsequent modules. Question Analysis task is responsible to clean the questions, i.e. removing question marks, interrogative pronoun and imperative verbs. Besides identifing each meaningful element of the question, the system annotate each word with their part-of-speech tag. The following information are set apart: main verb, question target, named entity.

The Passage Retrieval module is responsible for searching web pages using the keywords into the question and save them into local files for post-processing. This version of the system is working with 10 downloaded files.This module is also responsible for cleaning the HTML files and saving into local files only the content information. We use a relevance score based on the average of the number of words in the question to preserves the original idea of the question. After the content is saved into a file, the system will select the relevant sentences based on matching and counting the keywords in the sentences.

The Answer Extraction Module aims at identifying and extracting relevant answers and presents them in the form the a list. This module has two main tasks: Candidate Answer Identification and Building the List Answer. Candidate Answer Identification task extracts all words tagged with the proper name tag (in this version of the system version we are assuming that all answers are proper names).

The process of Building the List Answers based on frequency and word occur rules. The process based on frequency uses two lists: Premium List and Work List. The Premium List is composed by candidates extracted from sentences previously classified as highly relevant and will serve to guide the rest of the processing. If we were to consider only these elements, the list of answers would probably contain correct items. However, the list may be incomplete and lack elements. Then, continuing in the same vein of our strategy, the Work List is build with candidates extracted from sentences classified as medium and low. This list will be used to confirm and expand the elements in the list. The word occur rules take advantage of the title of web page, of sentences that perfectly matches with the question and of candidate verb that matches with the question verb.

4.1 Results

For the experiments we used a set of 10 questions[1] that require List Answers:
Q1: Instrumentos musicais de origem africana comuns no Brasil. *African musical instruments common in Brazil.*
Q2: Parques do Rio de Janeiro que têm cachoeiras. *Parks of Rio de Janeiro that have waterfalls.*
Q3: Igrejas em Macau. *Churches in Macau.*
Q4: Cidades que fizeram parte do domínio português na Índia. *Cities in India that were under Portuguese rule.*
Q5: Parques nacionais de Moçambique. *National parks in Mozambique*
Q6: Ilhas de Moçambique. *Islands of Mozambique*
Q7: Movimentos culturais surgidos no nordeste do Brasil. *Cultural movements that emerged n the northeast of Brazil.*
Q8: Dioceses católicas de Moçambique. *Catholic dioceses in Mozambique.*
Q9: Candidatos a alguma das eleições presidenciais na Guiné-Bissau. *Candidates for any of the presidential elections in Guinea-Bissau.*
Q10: Capitais das províncias de Angola. *Capitals of the provinces in Angola.*

[1]These questions were based on Pagico: www.linguateca.pt/Pagico

Table 2 shows the metric evaluation of the LX-ListQuestion. The metrics used are: recall, precision and F-measure. These metrics take into consideration two lists: a reference list (correct answers expected) and the system list (answers returned by the QA system). Precision: C is the number of common elements between reference and system lists and S is the number of elements given by the system.
$Precision = \frac{C}{S}$
Recall: C is the number of common elements between reference and system lists and L is the number of elements in reference list.
$Recall = \frac{C}{L}$
F-measure: its the combination between Recall and Precision.
$F - measure = \frac{2*Recall*Precision}{Recall+Precision}$

Table 2: Metric Evaluation.

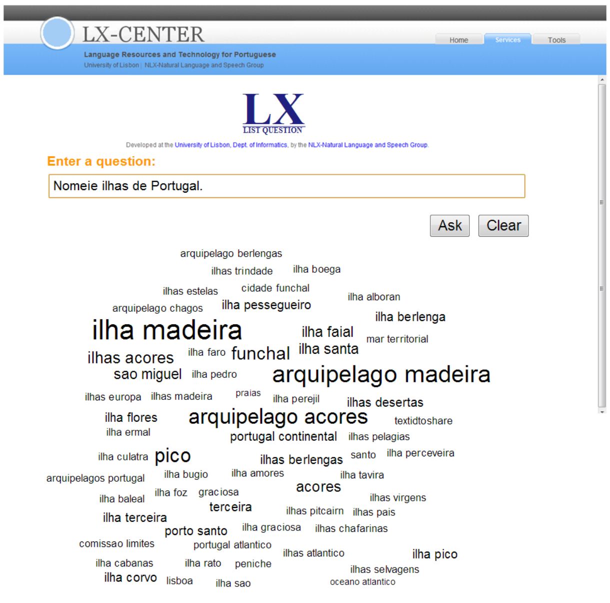
Question	Precision	Recall	F-Measure
Q1	0.15	0.36	0.21
Q2	0.08	0.50	0.14
Q3	0.13	0.35	0.18
Q4	0.14	0.23	0.17
Q5	0.10	0.75	0.18
Q6	0.13	0.42	0.20
Q7	0.05	0.20	0.08
Q8	0.15	0.71	0.24
Q9	0.05	0.25	0.09
Q10	0.38	0.42	0.40
AVERAGE	0.14	0.38	0.20

We observe from Table 2 that the system answered all questions. It achieved better recall for the questions Q5 and Q8. The Question Q10 obtained better precision and also f-measure. Overall, the system scores 0.38 of recall, which is a very competitive result for the current state-of-art. Exploring the redundancy of information seems to be a good approach to this task, but it alone cannot handle all problems. The word occur rules implemented was important step to enrich the system.

4.2 User interface

LX-ListQuestion is available on the web:
http://nlxserv.di.fc.ul.pt/lxlistquestion/
In our tests, the response time ranged between 16 and 28 seconds of processing from submitting the question and getting the list of answers. We chose to use word cloud instead of a traditional list as presentation of results because the final list an-

swers evidence not only the correct answers but of the possibly relevant words related to the question. The confidence that the system has a given answer is based on the frequency of words found in the texts collected from the web. The most frequent words are represented in the word cloud using a greater font size. The word cloud also helps the user to understand the context in which the answers may be embedded. Figure 2 shows LX-ListQuestion online GUI.



Nomeie ilhas de Portugal.

Figure 2: LX-ListQuestion online GUI

5 Concluding remarks

LX-ListQuestion is a fully-fledged Web based QA system that generates answers to list questions and presents them in a word cloud. The system exploits the redundancy of information available in the Web and combine with word occur rules to improve QA accuracy. This version handles Portuguese Language. The next version will be extended to provide answers to other languages as well. This work has being developed as the subject of a progressing doctoral thesis and new improvements will be implemented.

References

Nuno Cardoso, David Batista, Francisco J. López-Pellicer, and Mário J. Silva. 2009. Where in the wikipedia is that answer? the xldb at the gikiclef 2009 task. In Carol Peters, Giorgio Maria Di Nunzio, Mikko Kurimo, Djamel Mostefa, Anselmo Peñas, and Giovanna Roda, editors, *CLEF*, volume 6241 of *Lecture Notes in Computer Science*, pages 305–309. Springer.

Iustin Dornescu. 2009. Semantic qa for encyclopaedic questions: Equal in gikiclef. In Carol Peters, Giorgio Maria Di Nunzio, Mikko Kurimo, Djamel Mostefa, Anselmo Peñas, and Giovanna Roda, editors, *CLEF*, volume 6241 of *Lecture Notes in Computer Science*, pages 326–333. Springer.

Robert J. Gaizauskas, Mark A. Greenwood, Henk Harkema, Mark Hepple, Horacio Saggion, and Atheesh Sanka. 2005. The university of sheffield s trec 2005 q&a experiments. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, November 15-18, 2005*, volume Special Publication 500-266. National Institute of Standards and Technology (NIST).

Sven Hartrumpf and Johannes Leveling. 2010. GIRSA-WP at GikiCLEF: Integration of structured information and decomposition of questions. In *10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30-October 2, Revised Selected Papers*, Lecture Notes in Computer Science (LNCS). Springer. (to appear).

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Ying Shi, and Bryan Rink. 2006. Question answering with lcc's chaucer at trec 2006. In *TREC*.

Majid Razmara and Leila Kosseim. 2008. Answering list questions using co-occurrence and clustering. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*. European Language Resources Association.

Edward W. D. Whittaker, Josef R. Novak, Pierre Chatain, and Sadaoki Furui. 2006. Trec 2006 question answering experiments at tokyo institute of technology. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006*, volume Special Publication 500-272. National Institute of Standards and Technology (NIST).

Min Wu and Tomek Strzalkowski. 2006. Utilizing co-occurrence of answers in question answering. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.

Hui Yang and Tat-Seng Chua. 2004. Web-based list question answering. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1277, Morristown, NJ, USA. Association for Computational Linguistics.

Hui Yang, Hang Cui, Mstislav Maslennikov, Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2003. Qualifier in trec-12 qa main task. In *TREC*, pages 480–488.

Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework

S. Schlögl
MCI Management Center Innsbruck
Management, Communication & IT
Innsbruck, AUSTRIA
schlogl@mci.edu

P. Milhorat*, **G. Chollet***, **J. Boudy**†
Institut Mines-Télécom
*Télécom ParisTech & †Télécom SudParis
Paris, FRANCE
milhorat@telecom-paristech.fr

Abstract

Wizard of Oz (WOZ) prototyping employs a human wizard to simulate anticipated functions of a future system. In Natural Language Processing this method is usually used to obtain early feedback on dialogue designs, to collect language corpora, or to explore interaction strategies. Yet, existing tools often require complex client-server configurations and setup routines, or suffer from compatibility problems with different platforms. Integrated solutions, which may also be used by designers and researchers without technical background, are missing. In this paper we present a framework for multi-lingual dialog research, which combines speech recognition and synthesis with WOZ. All components are open source and adaptable to different application scenarios.

1 Introduction

In recent years Language Technologies (LT) such as Automatic Speech Recognition (ASR), Machine Translation (MT) and Text-to-Speech Synthesis (TTS) have found their way into an increasing number of products and services. Technological advances in the field have created new possibilities, and ubiquitous access to modern technology (i.e. smartphones, tablet computers, etc.) has inspired novel solutions in multiple application areas. Still, the technology at hand is not perfect and typically substantial engineering effort (gathering of corpora, training, tuning) is needed before prototypes involving such technologies can deliver a user experience robust enough to allow for potential applications to be evaluated with real users. For graphical interfaces, well-known prototyping methods like sketching and wire-framing allow for obtaining early impressions and initial user feedback. These low-fidelity prototyping techniques do not, however, work well with speech and natural language. The Wizard of Oz (WOZ) method can be employed to address this shortcoming. By using a human 'wizard' to mimic the functionality of a system, either completely or in part, WOZ supports the evaluation of potential user experiences and interaction strategies without the need for building a fully functional product first (Gould et al., 1983). It furthermore supports the collection of domain specific language corpora and the easy exploration of varying dialog designs (Wirén et al., 2007). WOZ tools, however, are often application dependent and built for very specific experimental setups. Rarely, are they re-used or adapted to other application scenarios. Also, when used in combination with existing technology components such as ASR or TTS, they usually require complex software installations and server-client configurations. Thus, we see a need for an easy 'out-of-the-box' type solution. A tool that does not require great technical experience and therefore may be used by researchers and designers outside the typical NLP research and development community. This demo is the result of our recent efforts aimed at building such an integrated prototyping tool.

We present a fully installed and configured server image that offers multi-lingual (i.e. English, German, French, Italian) ASR and TTS integrated with a web-based WOZ platform. All components are open-source (i.e. adaptable and extendable) and connected via a messaging server and a number of Java programs. When started the framework requires only one single script to be executed (i.e. there is a separate script for each language so that the components are started using the right parameters) in order to launch a WOZ driven system environment. With such a pre-configured setup we believe that also non-NLP experts are able to successfully conduct extended user studies for language technologies applications.

2 Existing Comparable Tools

Following the literature, existing tools and frameworks that support prototyping of language technology applications can be separated into two categories. The first category consists of so-called Dialogue Management (DM) tools, which focus on the evaluation of Language Technologies (LTs) and whose primary application lies in the areas of NLP and machine learning. Two well-known examples are the CSLU toolkit (Sutton et al., 1998) and the Olympus dialogue framework (Bohus et al., 2007). Others include the Jaspis dialogue management system (Turunen and Hakulinen, 2000) and the EPFL dialogue platform (Cenek et al., 2005). DM tools explore the language-based interaction between a human and a machine and aim at improving this dialogue. They usually provide an application development interface that integrates different LTs such as ASR and TTS, which is then used by an experimenter to specify a pre-defined dialogue flow. Once the dialogue is designed, it can be tested with human participants. The main focus of these tools lies on testing and improving the quality of the employed technology components and their interplay. Unlike DM tools, representatives from the second category, herein after referred to as WOZ tools, tend to rely entirely on human simulation. This makes them more interesting for early feedback, as they better support the aspects of low-fidelity prototyping. While these applications often offer more flexibility, they rarely integrate actual working LTs. Instead, a human mimics the functions of the machine, which allows for a less restrictive dialogue design and facilitates the testing of user experiences that are not yet supported by existing technologies. Most WOZ tools, however, should be categorized as throwaway applications i.e. they are built for one scenario and only rarely re-used in other settings. Two examples that allow for a more generic application are SUEDE (Klemmer et al., 2000) and Richard Breuer's WOZ tool[1].

While both DM and WOZ tools incorporate useful features, neither type provides a full range of support for low-fidelity prototyping of LT applications. DM tools lack the flexibility of exploring aspects that are currently not supported by technology, and pure WOZ applications often depend too much on the actions of the wizard, which can lead to unrealistic human-like behaviour and inconsistencies with its possible bias on evaluation results. A combination of both types of tools can outweigh their deficiencies and furthermore allow for supporting different stages of prototyping. That is, a wizard might complement existing technology on a continuum by first taking on the role of a 'controller' who simulates technology. Then, in a second stage one could act as a 'monitor' who approves technology output, before finally moving on to being a 'supervisor' who only overrides output in cases where it is needed (Dow et al., 2005). However, to allow for such variation an architecture is required that on the one hand supports a flexible use of technology components and on the other hand offers an interface for real-time human intervention.

3 Integrated Prototyping Framework

In order to offer a flexible and easy to use prototyping framework for language technology applications we have integrated a number of existing technology components using an Apache ACTIVEMQ messaging server[2] and several Java programs. Our framework consists of the JULIUS Large Vocabulary Continuous Speech Recognition engine[3], an implementation of the GOOGLE SPEECH API[4], the WEBWOZ Wizard of Oz Prototyping Platform[5] and the MARY Text-to-Speech Synthesis Platform[6]. All components are fully installed and connected running on a VIRTUAL BOX server image[7] (i.e. Ubuntu 12.04 LTS Linux Server). Using this configuration we offer a platform that supports real-time speech recognition as well as speech synthesis in English, French, German and Italian. Natural Language Understanding (NLU), Dialog Management (DM), and Natural Language Generation (NLG) is currently performed by the human 'wizard'. Respective technology components may, however, be integrated in future versions of the framework. The following sections describe the different components in some more detail and elaborate on how they are connected.

[1]http://www.softdoc.de/woz/index.html

[2]http://activemq.apache.org/

[3]http://julius.sourceforge.jp/en_index.php

[4]http://www.google.com/intl/en/chrome/demos/speech.html

[5]https://github.com/stephanschloegl/WebWOZ

[6]http://mary.dfki.de/

[7]https://www.virtualbox.org/

3.1 Automatic Speech Recognition

The JULIUS open-source Large Vocabulary Continuous Speech Recognition engine (LVCSR) uses n-grams and context-dependent Hidden Markov Models (HMM) to transform acoustic input into text output (Lee et al., 2008). Its recognition performance depends on the availability of language dependent resources i.e. acoustic models, language models, and language dictionaries. Our framework includes basic language resources for English, German, Italian and French. As those resources are still very limited we have also integrated online speech recognition for these four languages using the Google Speech API. This allows for conducting experiments with users while at the same time collecting the necessary data for augmenting and filling in JULIUS language resources.

3.2 Text-to-Speech Synthesis

MARY TTS is a state-of-the-art, open source speech synthesis platform supporting a variety of different languages and accents (Schröder and Trouvain, 2003). For the here presented multilingual prototyping framework we have installed synthesized voices for US English (cmu-slt-hsmm), Italian (istc-lucia-hsmm), German (dfki-pavoque-neutral) as well as French (enst-dennys-hsmm). Additional voices can be downloaded and added through the MARY component installer.

3.3 Wizard of Oz

WebWOZ is a web-based prototyping platform for WOZ experiments that allows for a flexible integration of existing LTs (Schlögl et al., 2010). It was implemented using modern web technologies (i.e. Java, HTML, CSS) and therefore runs in any current web browser. It usually uses web services to integrate a set of pre-configured LT components (i.e. ASR, MT, TTS). For the presented prototyping framework, however, we have integrated WebWOZ with our ASR solution (i.e. the combined Google/JULIUS engine) and MARY TTS. Consequently ASR output is displayed in the top area of the wizard interface. A wizard is then able to select an appropriate response from a set of previously defined utterances or use a free-text field to compose a response on the fly. In both cases the utterance is sent to the MARY TTS server and spoken out by the system.

3.4 Messaging Server and Gluing Programs

In order to achieve the above presented integration of ASR, WOZ and TTS we use an Apache ACTIVEMQ messaging server and a number of Java programs. One of these programs takes the output from our ASR component and inserts it into the WebWOZ input stream. In addition it publishes this output to a specific ASR ActiveMQ queue so that other components (e.g. potentially an NLU component) may also be able to process it. Once an ASR result is available within WebWOZ, it is up to the human wizard to respond. WebWOZ was slightly modified so that wizard responses are not only sent to the internal WebWOZ log, but also to a WIZARD ActiveMQ queue. A second Java program then takes the wizard responses from the WIZARD queue and pushes them to a separate MARY queue. While it may seem unnecessary to first take responses from one queue just to publish them to another queue, it allows for the easy integration of additional components. For example, we have also experimented with a distinct NLG component. Putting this component between the WIZARD and the MARY queue we were able to conduct experiments where a wizard instead of sending entire text utterance would rather send text-based semantic frames (i.e. a semantically unified representation of a user's input). Such shows the flexibility of using the described queue architecture. Finally we use a third Java program to take text published to the MARY queue (i.e. either directly coming from the wizard or produced by an NLG component as with one of our experimental settings) and send it to the MARY TTS server. Figure 1 illustrates the different framework components and how they are connected to each other.

4 Demo Setup

The optimal setup for the demo uses two computer stations, one for a wizard and one for a test user. The stations need to be connected via a LAN connection. The test user station runs the prototyping framework, which is a fully installed and configured Virtual Box software image (Note: any computer capable of running Virtual Box can serve as a test user station). The wizard station only requires a modern web browser to interact with the test user station. A big screen size (e.g. 17 inch) for the wizard is recommended as such eases his/her task. Both stations will be provided by the authors.

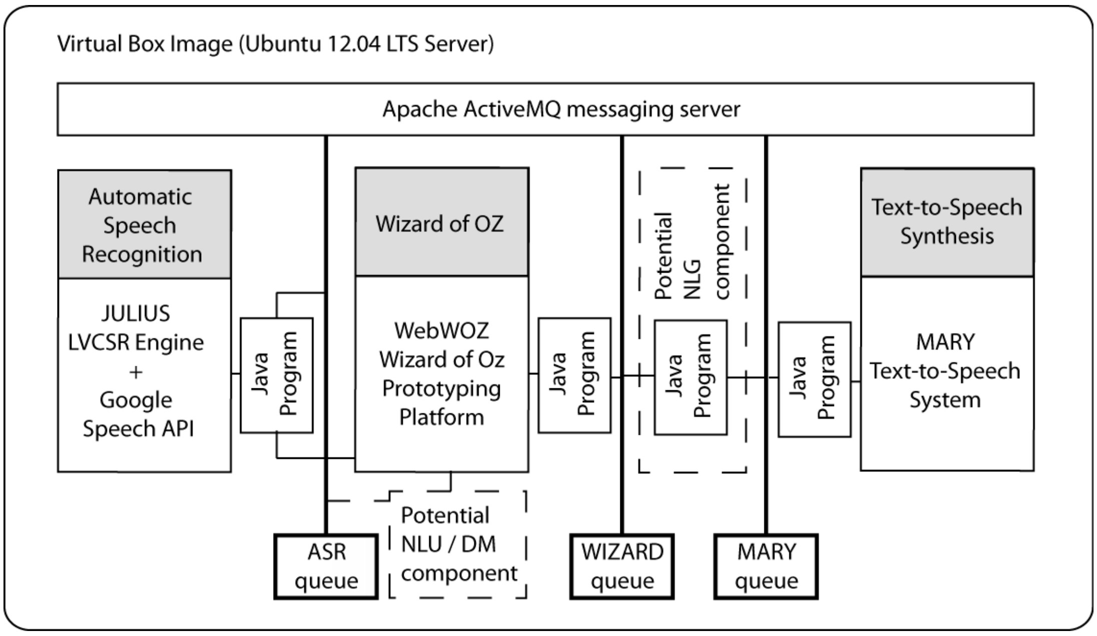

Virtual Box Image (Ubuntu 12.04 LTS Server)
Apache ActiveMQ messaging server
Automatic Speech Recognition
JULIUS LVCSR Engine + Google Speech API
Java Program
Wizard of OZ
WebWOZ Wizard of Oz Prototyping Platform
Java Program
Potential NLG component
Java Program
Java Program
Text-to-Speech Synthesis
MARY Text-to-Speech System
ASR queue
Potential NLU / DM component
WIZARD queue
MARY queue

Figure 1: Prototyping Framework Components.

5 Summary and Future Work

This demo presents an integrated prototyping framework for running WOZ driven language technology application scenarios. Gluing together existing tools for ASR, WOZ and TTS we have created an easy to use environment for spoken dialog design and research. Future work will focus on adding additional language technology components (e.g. NLU, DM, NLG) and on improving the currently limited ASR language resources.

Acknowledgments

The presented research is conducted as part of the vAssist project (AAL-2010-3-106), which is partially funded by the European Ambient Assisted Living Joint Programme and the National Funding Agencies from Austria, France and Italy.

References

D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. 2007. Olympus: An open-source framework for conversational spoken language interface research. In *Proc. of NAACL-HLT*, pages 32–39.

P. Cenek, M. Melichar, and M. Rajman. 2005. A framework for rapid multimodal application design. In *Proceedings of TSD*, pages 393–403.

S. Dow, B. Macintyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy. 2005. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26.

J. D. Gould, J. Conti, and T. Hovanyecz. 1983. Composing letters with a simulated listening typewriter. *Communications of the ACM*, 26(4):295–308.

S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. 2000. SUEDE: A wizard of oz prototyping tool for speech user interfaces. In *Proc. of UIST*, pages 1–10.

C. Lee, S. Jung, and G. G. Lee. 2008. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Proc. of ACL-HLT*, pages 630–637.

S. Schlögl, G. Doherty, N. Karamanis, and S Luz. 2010. WebWOZ: a wizard of oz prototyping framework. In *Proc. of the ACM EICS Symposium on Engineering Interactive Systems*, pages 109–114.

M. Schröder and J. Trouvain. 2003. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*.

S. Sutton, R. Cole, J. de Vielliers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, D. Massaro, and M. Cohen. 1998. Universal speech tools: The CSLU toolkit.

M. Turunen and J. Hakulinen. 2000. Jaspis- a framework for multilingual adaptive speech applications. In *Proc. of ICSLP*, pages 719–722.

M. Wirén, R. Eklund, F. Engberg, and J. Westermark. 2007. Experiences of an In-Service Wizard-of-Oz Data Collection for the Deployment of a Call-Routing Application. In *Proc. of NAACL-HLT*, pages 56–63.

RelationFactory: A Fast, Modular and Effective System for Knowledge Base Population

Benjamin Roth[†] **Tassilo Barth**[†] **Grzegorz Chrupała**[*] **Martin Gropp**[†] **Dietrich Klakow**[†]
[†]Spoken Language Systems, Saarland University, 66123 Saarbrücken, Germany
[*]Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands
[†]{beroth|tbarth|mgropp|dietrich.klakow}@lsv.uni-saarland.de
[*]g.chrupala@uvt.nl

Abstract

We present *RelationFactory*, a highly effective open source relation extraction system based on shallow modeling techniques. *RelationFactory* emphasizes modularity, is easily configurable and uses a transparent pipelined approach.

The interactive demo allows the user to pose queries for which *RelationFactory* retrieves and analyses contexts that contain relational information about the query entity. Additionally, a recall error analysis component categorizes and illustrates cases in which the system missed a correct answer.

1 Introduction and Overview

Knowledge base population (KBP) is the task of finding relational information in large text corpora, and structuring and tabularizing that information in a knowledge base. Given an entity (e.g. of type `PERSON`) with an associated relational schema (a set of relations, e.g. `city_of_birth(PERSON, CITY)`, `schools_attended(PERSON, ORGANIZATION)`, `spouse(PERSON, PERSON)`), all relations about the entity that are expressed in a text corpus would be relevant, and the correct answers would have to be extracted.

The TAC KBP benchmarks[1] are an effort to formalize this task and give researchers in the field the opportunity to evaluate their algorithms on a set of currently 41 relations. In TAC KBP, the task and evaluation setup is established by well-defined information needs about query entities of types `PERSON` and `ORGANIZATION` (e.g. who is the `spouse` of a person, how many `employees` does an organization have). A perfect system would have to return all relevant information (and only this) contained in the text corpus. TAC KBP aims at giving a realistic picture of not only precision but also recall of relation extraction systems on big corpora, and is therefore an advancement over many other evaluations done for relation extraction that are often precision oriented (Suchanek et al., 2007) or restrict the gold key to answers from a fixed candidate set (Surdeanu et al., 2012) or to answers contained in a data base (Riedel et al., 2010). Similar to the classical TREC evaluation campaigns in document retrieval, TAC KBP aims at approaching a true recall estimate by pooling, i.e. merging the answers of a timed-out manual search with the answers of all participating systems. The pooled answers are then evaluated by human judges.

It is a big advantage of TAC KBP that the end-to-end setup (from the query, through retrieval of candidate contexts and judging whether a relation is expressed, to normalizing answers and putting them into a knowledge base) is realistic. At the same time, the task is very complex and may involve too much work overhead for researchers only interested in a particular step in relation extraction such as matching and disambiguation of entities, or judging relational contexts. We therefore introduce *RelationFactory*, a fast, modular and effective relation extraction system, to the research community as open source software.[2] *RelationFactory* is based on distantly supervised classifiers and patterns (Roth et al., 2013), and was top-ranked (out of 18 systems) in the TAC KBP 2013 English Slot-filling benchmark (Surdeanu, 2013).

In this demo, we give potential users the possibility to interact with the system and to get a feel for use cases, strengths and limitations of the current state of the art in knowledge base population.

[1]`http://www.nist.gov/tac/about/`

[2]`https://github.com/beroth/relationfactory`

The demo illustrates how *RelationFactory* arrives at its conclusions and where future potentials in relation extraction lie. We believe that *RelationFactory* provides an easy start for researchers interested in relation extraction, and we hope that it may serve as a baseline for new advances in knowledge base population.

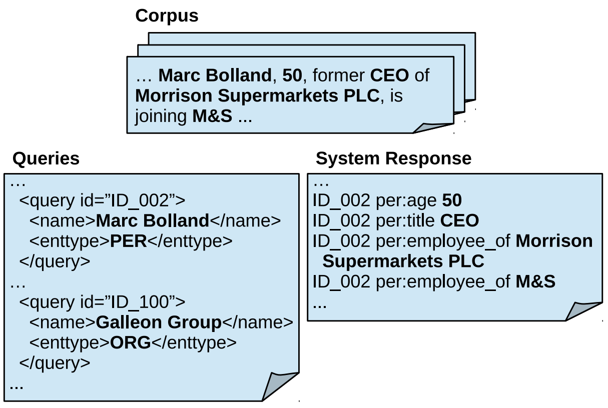
2 System Philosophy and Design Principles

The design principles of *RelationFactory* conform to what is known as the *Unix philosophy*.[3] For *RelationFactory* this philosophy amounts to a set of modules that solve a certain step in the pipeline and can be run (and tested) independently of the other modules. For most modules, input and output formats are column-based text representations that can be conveniently processed with standard Linux tools for easy diagnostics or prototyping. Data representation is compact: the system is designed in a way that each module ideally outputs one new file. Because of modularization and simple input and output formats, *RelationFactory* allows for easy extensibility, e.g. for research that focuses solely on novel algorithms at the prediction stage.

The single modules are connected by a makefile that controls the data flow and allows for easy parallelization. *RelationFactory* is highly configurable: new relations can be added without changing any of the source code, only by changing configuration files and adding or training respective relational models.

Furthermore, *RelationFactory* is designed to be highly scalable: Thanks to feature hashing, large amounts of training data can be used in a memory-friendly way. Predicting relations in real-time is possible using shallow representations. Surface patterns, ngrams and skip-ngrams allow for highly accurate relational modeling (Roth et al., 2013), without incurring the cost of resource-intensive processing, such as parsing.

[3]One popular set of tenets (Gancarz, 2003) summarizes the *Unix philosophy* as:
1. Small is beautiful.
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency.
5. Store data in flat text files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase leverage and portability.
8. Avoid captive user interfaces.
9. Make every program a filter.



Corpus
… Marc Bolland, 50, former CEO of Morrison Supermarkets PLC, is joining M&S ...
Queries
…
<query id="ID_002">
<name>Marc Bolland</name>
<enttype>PER</enttype>
</query>
…
<query id="ID_100">
<name>Galleon Group</name>
<enttype>ORG</enttype>
</query>
…
System Response
…
ID_002 per:age 50
ID_002 per:title CEO
ID_002 per:employee_of Morrison Supermarkets PLC
ID_002 per:employee_of M&S
…

Figure 1: TAC KBP: Given a set of queries, return a correct, complete and non-redundant response with relevant information extracted from the text corpus.

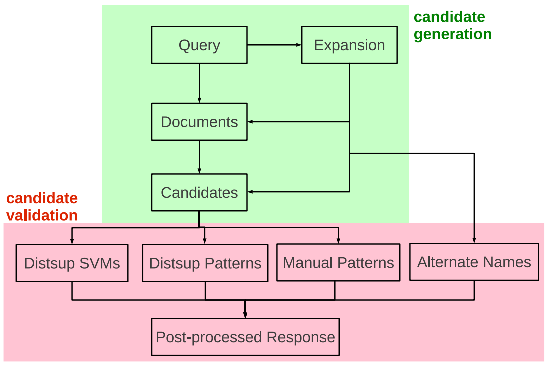


Figure 2: Data flow of the relation extraction system: The *candidate generation* stage retrieves possible relational contexts. The *candidate validation* stage predicts whether relations actually hold and produces a valid response.

3 Component Overview

A simplified input and output to *RelationFactory* is shown in Figure 1. In general, the pipeline is divided in a *candidate generation* stage, where documents are retrieved and candidate sentences are identified, and the *candidate validation* stage, which predicts and generates a response from the retrieved candidates (see Figure 2).

In a first step, the system generates aliases for the query using statistical and rule-based expansion methods, for example:

Query	Expansion
Adam Gadahn	Azzam the American, Adam Yahiye Gadahn, Gadahn
STX Finland	Kvaerner Masa Yards, Aker Finnyards, STX Finland Ltd

The expansions are used for retrieving documents from a Lucene index. All those sentences are retained where the query (or one of the query aliases) is contained and the named-entity tagger has identified another entity with the type of a potential answer for one of the sought relations. The system is easily configurable to include matching of non-standard named-entity types from lists. *RelationFactory* uses lists obtained from Freebase (`www.freebase.com`) to match answer candidates for the types `CAUSE-OF-DEATH`, `JOB-TITLE`, `CRIMINAL-CHARGES` and `RELIGION`.

The candidate sentences are output line-by-line and processed by one of the *validation modules*, which determine whether actually one of the relations is expressed. *RelationFactory* currently uses three standard validation modules: One based on SVM classifiers, one based on automatically induced and scored patterns, and one based on manually crafted patterns. The validation modules function as a filter to the candidates file. They do not have to add a particular formatting or conform to other requirements of the KBP task such as establishing non-redundancy or finding the correct offsets in the text corpus. This is done by other modules in the pipeline, most notably in the post-processing step, where statistical methods and heuristics are applied to produce a well-formed TAC KBP response.

4 User Perspective

From a user perspective, running the system is as easy as calling:

```
./run.sh system.config
```

The configuration file contains all information about the general run configuration of the system, such as the query file to use, the format of the response file (e.g. TAC 2012 or TAC 2013 format), the run directory that will contain the response, and the Lucene index with the corpus. Optional configuration can control non-standard validation modules, and special low or high-recall query expansion schemes.

The relevant parts of the configuration file for a standard 2013 TAC KBP run would look like the following:

```
query /TAC_EVAL/2013/query.xml
goal response2013
rundir /TAC_RUNS/run2013/
index /TAC_CORPORA/2013/index
rellist /CFG/rellist2013
relations.config /CFG/relations2013.config
```

The last two lines refer to relation-specific configuration files: The list of relations to use and information about them. Changing these files (and adding respective models) allows for inclusion of further relations. The relation-specific configuration file contains information about the query entity type, the expected answer named-entity tag and whether a list of answers is expected (compared to relations with just one correct answer):

```
per:religion enttype PER
per:religion argtag RELIGION
per:religion listtype false
org:top_members_employees enttype ORG
org:top_members_employees argtag PERSON
org:top_members_employees listtype true
```

RelationFactory comes with batteries included: The models and configurations for TAC KBP 2013 work out-of-the-box and can easily be used as a relation extraction module in a bigger setting or as a baseline for new experiments.[4]

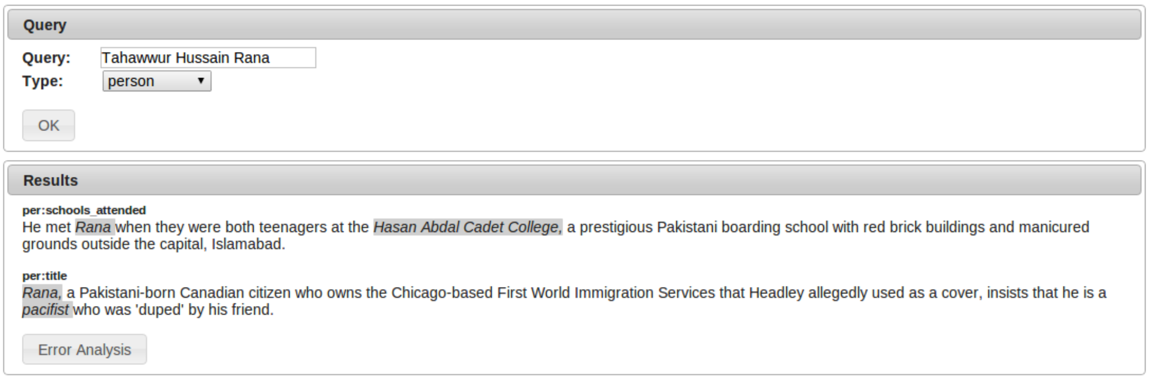
5 Illustrating *RelationFactory*

In TAC KBP 2013, 6 out of 18 systems achieved an F1 score of over 30%. *RelationFactory* as the top-performing system achieved 37.28% compared to 68.49% achieved by human control annotators (Surdeanu, 2013). These numbers clearly show that current systems have just gone halfway toward achieving human-like performance on an end-to-end relation extraction task.

The aim of the *RelationFactory* demo is to illustrate what the current challenges in TAC KBP are. The demonstration interface therefore not only shows the answers generated for populating a potential knowledge base, but also what text was used to justify the extraction.

The real-time performance of *RelationFactory* allows for trying arbitrary queries and changing the configuration files and immediately seeing the effects. Different expansion schemes, validation modules and patterns can be turned on and off, and intuitions can be obtained about the bottlenecks and error sources of relation extraction. The demo also allows for seeing the effect of extracting information from different corpora: a Wikipedia corpus and different TAC KBP corpora, such as newswire and web text.

[4]Training models for new relations requires is a bigger effort and includes generation of distant supervision training data by getting argument pairs from relational patterns or a knowledge base like Freebase. *RelationFactory* includes some training scripts but since they are typically run once only, they are significantly less documented.



Tahawwur Hussain Rana
person

per:schools_attended
He met Rana when they were both teenagers at the Hasan Abdal Cadet College, a prestigious Pakistani boarding school with red brick buildings and manicured grounds outside the capital, Islamabad.

per:title
Rana, a Pakistani-born Canadian citizen who owns the Chicago-based First World Immigration Services that Headley allegedly used as a cover, insists that he is a pacifist who was 'duped' by his friend.

Figure 3: Screenshot of the *RelationFactory* demo user interface.

RelationFactory contains a number of diagnostic tools: With a gold key for a set of queries, error classes can be broken down and examples for certain error classes can be shown. For example, the diagnostic tool for missed recall performs the following checks:

1. **Is document retrieved?**
2. **Is query matched?** This determines whether a sentence is considered for further processing.
3. **Is answer in query sentence?** Whether the answer is in one of the sentences with the query. Our system only can find answers when this is the case, as there is no coreference module included.
4. **Do answer tags overlap with gold answer?**
5. **Do they overlap exactly?**
6. **Other (validation).** If all previous checks are passed, the candidate has correctly been generated by the candidate generation stage, but the validation modules have failed to predict the relation.

On the TAC KBP 2013 queries, the resulting recall error analysis is:

error class	missing recall
Doc not retrieved	5.59%
Query not matched	10.37%
Answer not in query sentence	16.63%
Answer tag inexact	5.36%
Answer not tagged	24.85%
Other (validation)	37.17%

The demonstration tool allows for inspection of instances of each of the error classes.

6 Conclusion

This paper illustrates *RelationFactory*, a modular open source knowledge-base population system. We believe that *RelationFactory* will become especially valuable for researchers in the field of relation extraction that focus on one particular problem of knowledge-base-population (such as entity expansion or relation prediction) and want to integrate their algorithms in an end-to-end setting.

Acknowledgments

Benjamin Roth is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship. Tassilo Barth was supported in part by IARPA contract number W911NF-12-C-0015.

References

Mike Gancarz. 2003. *Linux and the Unix philosophy*. Digital Press.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL)*, pages 455–465. ACL.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

MMAX2 for coreference annotation

Mateusz Kopeć
Institute of Computer Science, Polish Academy of Sciences,
Jana Kazimierza 5, 01-248 Warsaw, Poland
m.kopec@ipipan.waw.pl

Abstract

This article presents major modifications in the MMAX2 manual annotation tool, which were implemented for the coreference annotation of Polish texts. Among other, a new feature of adjudication is described, as well as some general insight into the manual annotation tool selection process for the natural language processing tasks.

1 Introduction

Recently published Polish Coreference Corpus (PCC) (Ogrodniczuk et al., 2013) contains a large number of Polish texts annotated manually with coreference. During the initial stage of this project in 2011, a tool had to be selected for the manual text annotation with coreference.

First issue considered during the selection was the alternative of desktop versus online annotation tool. Recently, online annotation tools are becoming increasingly popular (see for example BRAT (Stenetorp et al., 2012)), with their advantages such as the possibility to monitor the current state of annotation and make changes to the annotation tool easily, without the need to communicate with the annotators. However, in our opinion the choice should be made mainly based on the annotators' preferences, as their work efficiency is crucial for the cost of the task.

10 linguists (which were annotators in previous projects conducted by the authors of this paper) were asked in anonymous survey to choose one of following three options: 1) that they prefer an online annotation tool (not requiring any installation), 2) a desktop tool with the possibility to work without constant internet access, 3) that they do not have preference. Only one person chose online tool and one person chose the third option, leaving no choice for the annotation task organizers other than to prepare a desktop application. The drawback of this approach was the need to manage text distribution among the annotators, as all the work was done on local computers. Distribution was controlled by the DistSys application, available at the webpage zil.ipipan.waw.pl/DistSys.

After some analysis of the features required by the project's scope, the choice was narrowed to only two tools: MMAX2 (Müller and Strube, 2006) and Palinka (Orăsan, 2003). The problem with the latter was that is was not error-prone, and lack of publicly available sources did not allow to make project-specific corrections. Therefore MMAX2 environment was chosen for the manual coreference annotation. It is a general purpose cross-platform desktop annotation tool (written in Java), created to be configurable for many natural language annotation efforts. It's source is publicly available at the webpage mmax2.net. MMAX2 interface consists of the main window with the text being annotated (see for example figure 5) and several smaller windows facilitating various tasks (all the other figures).

2 Annotation scope

The annotation scope of the Polish Coreference Corpus project consisted of several subtasks for an annotator to perform for each text. Because the automatic preannotation was used, annotator's job consisted not only of addition of new markables, but also removal and correction of existing ones. Subtasks to perform were to:

- mark all mentions,
- mark each mention's semantic head (choose one word the mention consists of),
- cluster coreferent mentions,
- mark dominant expression of each cluster,

- mark quasi-identity links.

MMAX2 was easy to configure for most of the subtasks, except the dominant expressions (there is no possibility to define attributes of clusters, only markables) and the choice of semantic head (available types of attributes did not include a possibility to define a mention-dependent attribute).

Because an estimate of inter-annotator agreement had to be calculated for the corpus, some texts were annotated independently by two annotators. Agreement measures were then calculated, but as single gold standard annotation was needed for the corpus, they also had to be merged into single version by the adjudicator. This feature was also not present in MMAX2.

3 New features

Even with it's great number of features, there was no possibility to use MMAX2 without any changes in it's source code. Some changes were the result of project's scope requirements, some were added in response to annotator requests. New implemented features include:

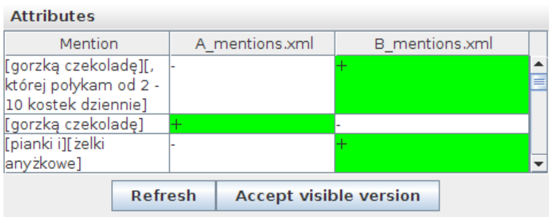
1. Internationalization – the interface of the tool is available in English and Polish and can be easily translated to other languages. Polish version was used by the annotators, but for international articles about the tool (such as this one) the English interface was used.

2. Semantic head selection – a dropdown list allows to choose one of the tokens mention consists of as it's semantic head. This head is also underlined in markable browser.

3. Storing user setting – which windows are opened, where are they located, what is the font and it's size – these and other user settings are saved and automatically restored when the application is reopened.

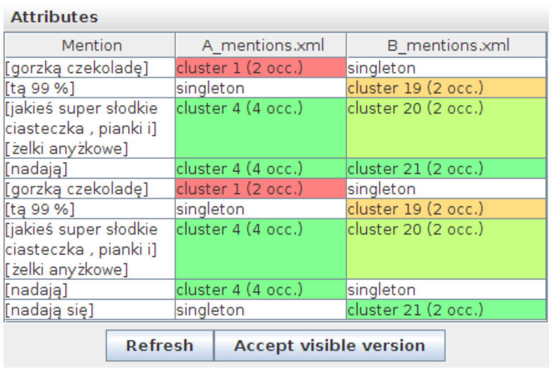
4. Dominant expressions – clusters can have their attribute: a dominant expression, which can be selected as one of the mentions from the cluster or any other expression entered by the user.

5. Undo button, reverting last action – very useful feature to revert the last change, regardless of it's nature.


Attributes

Mention	A_mentions.xml	B_mentions.xml
[gorzką czekoladę][, której połykam od 2 - 10 kostek dziennie]	-	+
[gorzką czekoladę]	+	-
[pianki i][żelki anyżkowe]	-	+

Figure 1: Adjudication of mentions


Attributes

Mention	A_mentions.xml	B_mentions.xml
[gorzką czekoladę]	cluster 1 (2 occ.)	singleton
[tą 99 %]	singleton	cluster 19 (2 occ.)
[jakieś super słodkie ciasteczka , pianki i] [żelki anyżkowe]	cluster 4 (4 occ.)	cluster 20 (2 occ.)
[nadają]	cluster 4 (4 occ.)	cluster 21 (2 occ.)
[gorzką czekoladę]	cluster 1 (2 occ.)	singleton
[tą 99 %]	singleton	cluster 19 (2 occ.)
[jakieś super słodkie ciasteczka , pianki i] [żelki anyżkowe]	cluster 4 (4 occ.)	cluster 20 (2 occ.)
[nadają]	cluster 4 (4 occ.)	singleton
[nadają się]	singleton	cluster 21 (2 occ.)

Figure 2: Adjudication of clustering

6. Merge two mentions – user can merge two mentions into one with a single click, summing their words and merging their clusters. Very useful feature when for example one is correcting automatic annotation, which failed to recognize a long named entity name and instead created two entities, each in it's separate cluster.

7. Improved browser operability – browsers allow to operate on mentions, links and clusters, not only to view them.

8. Adjudication feature – it will be covered in detail in the next section.

4 Adjudication feature

Adjudication feature of the new *Superannotation plugin* allows to compare two versions of annotation of the same text and merge them into one, adjudicated version. The design is based on the original MMAX2 Diff plugin, which allowed to see the differences between two annotations, yet it was not possible to merge them into one. The readability of the differences was also limited and it was improved in our tool.

The adjudication process starts with opening one annotation in standard way and then the other via the menu in *Superannotation plugin* and consist of several steps, each merging particular layer:

1. Mentions – first we need to merge mention annotations. Differences between the two annotations are shown in the figure 1. First column shows the mention content (and this is constant in all steps of adjudication), second shows if that mention is in the first annotation, third column shows if it is in the second annotation ("+" if yes, "-" if not). Single click at "+" or the first column highlights given span in the main window. Double click at one of the last two columns selects the clicked version as the proper one and changes the annotation in the other file to match the clicked version. After such double click, the difference disappears and that row vanishes. After all rows from that step are gone, mention annotations in both versions are the same and we can proceed to the next step.

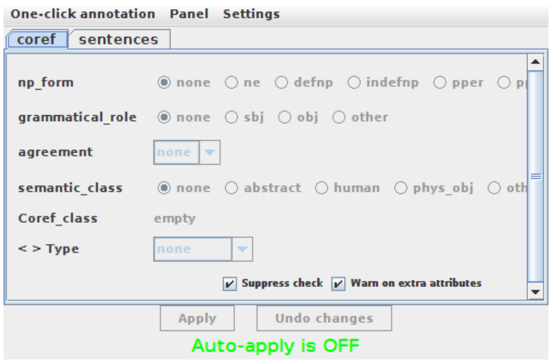
2. Comments – this time first column again shows each mention, for which there is a difference in comments in both annotations. Double clicking at 2nd or 3rd column resolves the difference in given row.

3. Heads – similar to comments, by double-clicking we can adjudicate differences in head annotation.

4. Links – analogously as with heads, we merge near-identity links annotations.

5. Clusters – this is the most complex adjudication task. At this point we surely have the same set of mentions in both annotations, but they may be clustered differently. Figure 2 presents how differences in clustering are visualized. Mentions with the same color in one column are in the same cluster (they have also the same cluster number). For example, two occurrences of mention *gorzką czekoladę* are in the same cluster according to the first annotation, and are singletons according to the second annotation. Single click on any of these options will show it in the main application window, while double click will choose the clicked version as the gold one and update the other to match it.

6. Dominating expressions – as the clusters are now the same, the only annotation left considers cluster attributes: dominating expressions.



np_form [✓ none] ne defnp indefnp pper p

grammatical_role [✓ none] sbj obj other

agreement none

semantic_class [✓ none] abstract human phys_obj oth

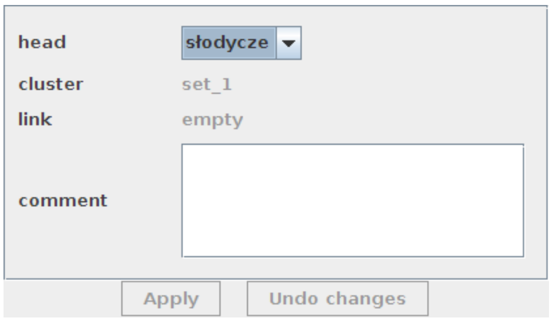
Coref_class empty

< > Type none

[✓ Suppress check] [✓ Warn on extra attributes]

Auto-apply is OFF

Figure 3: Mention attributes – original MMAX2



head słodycze

cluster set_1

link empty

comment

Figure 4: Mention attributes – simplified

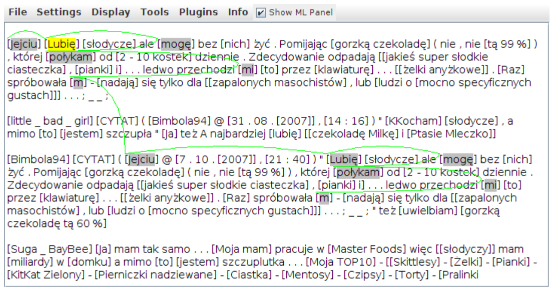
Key point of the adjudication procedure is to merge all differences at a given level before proceeding to the next one. This way, after we resolve all differences in the dominating expressions, we are certain that our annotations are fully merged and in fact the same.

5 Removed features

Because MMAX2 is a generic tool, the first impression is that it is very complicated. Numerous options, many windows and menus are overwhelming and could increase the time of creating a manual for the annotators (often writing a lot of text only to inform which options should not be changed). Therefore we removed many options and simplified the interface to leave only the features required by our annotation task. Compare for example the mention attribute window from the original MMAX2 in figure 4 and in our version in figure 3. Removed features included:

- Distinction of multiple annotation levels – scope of the project considers only one level, and the need to explicitly select it in many places (for example in markable browser) is unnecessary.

- Possibility to edit the base text – as we per-



[Jejciu] [Lubię] [słodycze] ale [mogę] bez [nich] żyć . Pomijając [gorzką czekoladę] (nie , nie [tą 99 %]) , której [połykam] od [2 - 10 kostek] dziennie . Zdecydowanie odpadają [[jakieś super słodkie ciasteczka] , [pianki] i] . . . ledwo przechodzi [mi] [to] przez [klawiaturę] . . . [[żelki anyżkowe]] . [Raz] spróbowała [m] - [nadają] się tylko dla [[zapalonych masochistów] , lub [ludzi o [mocno specyficznych gustach]]] . . . ; _ _ ;

[little _ bad _ girl] [CYTAT] ([Bimbola94] @ [31 . 08 . [2007]] , [14 : 16]) " [KKocham] [słodycze] , a mimo [to] [jestem] szczupła " [Ja] też A najbardziej [lubię] [[czekoladę Milkę] i [Ptasie Mleczko]]

[Bimbola94] [CYTAT] ([Jejciu] @ [7 . 10 . [2007]] , [21 : 40]) " [Lubię] [słodycze] ale [mogę] bez [nich] żyć . Pomijając [gorzką czekoladę] (nie , nie [tą 99 %]) , której [połykam] od [2 - 10 kostek] dziennie . Zdecydowanie odpadają [[jakieś super słodkie ciasteczka] , [pianki] i] . . . ledwo przechodzi [mi] [to] przez [klawiaturę] . . . [[żelki anyżkowe]] . [Raz] spróbowała [m] - [nadają] się tylko dla [[zapalonych masochistów] , lub [ludzi o [mocno specyficznych gustach]]] . . . ; _ _ ; " też [uwielbiam] [gorzką czekoladę tą 60 %]

[Suga _ BayBee] [Ja] mam tak samo . . . [Moja mam] pracuje w [Master Foods] więc [[słodyczy]] mam [miliardy] w [domku] a mimo [to] [jestem] szczuplutka . . . [Moja TOP10] - [[Skittlesy] - [Żelki] - [Pianki] - [KitKat Zielony] - [Pierniczki nadziewane] - [Ciastka] - [Mentosy] - [Czipsy] - [Torty] - [Pralinki

Figure 5: Unnecessary arcs

formed the inter-annotator agreement analysis, the base text could not be changed.

- Arcs between coreferent mentions in cluster (see figure 5 for original visualization) – from our experience, they decrease the readability of the cluster annotation. As the mentions in cluster are already highlighted, there is no need to show the arcs connecting them (the arcs are not clickable as in BRAT).

- MMAX Query Language – MMAX2 facilitates a query language to search for the annotations fulfilling given properties. In our opinion this feature seems more appropriate for an analyst, not an annotator. Moreover, results of such querying would be more informative for a collection of texts, not a single document.

- Kappa statistic and coincidence matrix calculation for multiple annotations of a single text – again, this feature seems more appropriate for an analyst and for the whole corpus, not a single text.

6 Conclusion

Every unnecessary action, which has to be repeated numerous times by a human annotator, has a significant cost in terms of time and money. We claim that annotation efforts are more efficient when there is a step of tool customization (or even design and implementation from scratch) beforehand and also during the process, based on the feedback from the annotators. Using general-purpose tools has a clear benefit of cheap and fast initialization of the project, but also there are major drawbacks: a compromise between the project needs and the tool capabilities. As we have seen, even a tool with great customization options such as MMAX2 doesn't have all the features one would need.

Experience of the PCC project shows, that instead of trying to provide a general, configurable annotation tool (which is very complex due to its wide application possibilities), another way to proceed is to create simple, well designed tool focused on specific task. Such tool can be then customized or extended by qualified programmers without much effort and then provide great efficiency of the annotation process.

Presented version of MMAX2 with its source code is available at `http://zil.ipipan.waw.pl/MMAX4CORE` webpage. We encourage it's use and modification for other coreference annotation projects.

Acknowledgments

The work reported here was cofounded by the Computer-based methods for coreference resolution in Polish texts project financed by the Polish National Science Centre (contract number 6505/B/T02/2011/40) and by the European Union from resources of the European Social Fund. Project PO KL „Information technologies: Research and their interdisciplinary applications".

References

Christoph Müller and Michael Strube. 2006. Multi-level annotation of linguistic data with mmax2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany.

Maciej Ogrodniczuk, Katarzyna Głowińska, Mateusz Kopeć, Agata Savary, and Magdalena Zawisławska. 2013. Polish coreference corpus. In Zygmunt Vetulani, editor, *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 494–498, Poznań, Poland. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. Adama Mickiewicza.

Constantin Orăsan. 2003. PALinkA: a highly customizable tool for discourse annotation. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialog*, pages 39 – 43, Sapporo, Japan, July, 5 -6.

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, Avignon, France, April. Association for Computational Linguistics.

The GATE Crowdsourcing Plugin: Crowdsourcing Annotated Corpora Made Easy

Kalina Bontcheva, Ian Roberts, Leon Derczynski, Dominic Rout
University of Sheffield
{kalina,ian,leon,d.rout}@dcs.shef.ac.uk

Abstract

Crowdsourcing is an increasingly popular, collaborative approach for acquiring annotated corpora. Despite this, reuse of corpus conversion tools and user interfaces between projects is still problematic, since these are not generally made available. This demonstration will introduce the new, open-source GATE Crowdsourcing plugin, which offers infrastructural support for mapping documents to crowdsourcing units and back, as well as automatically generating reusable crowdsourcing interfaces for NLP classification and selection tasks. The entire workflow will be demonstrated on: annotating named entities; disambiguating words and named entities with respect to DBpedia URIs; annotation of opinion holders and targets; and sentiment.

1 Introduction

Annotation science (Hovy, 2010; Stede and Huang, 2012) and general purpose corpus annotation tools (e.g. Bontcheva et al. (2013)) have evolved in response to the need for creating high-quality NLP corpora. Crowdsourcing is a popular collaborative approach that has been applied to acquiring annotated corpora and a wide range of other linguistic resources (Callison-Burch and Dredze, 2010; Fort et al., 2011; Wang et al., 2012). Although the use of this approach is intensifying, especially paid-for crowdsourcing, the reuse of annotation guidelines, task designs, and user interfaces between projects is still problematic, since these are generally not made available, despite their important role in result quality (Khanna et al., 2010).

A big outstanding challenge for crowdsourcing projects is that the cost to define a single annotation task remains quite substantial. This demonstration will introduce the new, open-source GATE Crowdsourcing plugin, which offers infrastructural support for mapping documents to crowdsourcing units, as well as automatically generated, reusable user interfaces[1] for NLP classification and selection tasks. Their use will be demonstrated on annotating named entities (selection task), disambiguating words and named entities with respect to DBpedia URIs (classification task), annotation of opinion holders and targets (selection task), as well as sentiment (classification task).

2 Crowdsourcing Stages and the Role of Infrastructural Support

Conceptually, the process of crowdsourcing annotated corpora can be broken down into four main stages, within which there are a number of largely infrastructural steps. In particular, data preparation and transformation into CrowdFlower units, creation of the annotation UI, creation and upload of gold units for quality control, and finally mapping judgements back into documents and aggregating all judgements into a finished corpus.

The rest of this section discusses in more detail where reusable components and infrastructural support for automatic data mapping and user interface generation are necessary, in order to reduce the overhead of crowdsourcing NLP corpora.

2.1 Project Definition

An important part of project definition is the mapping of the NLP problem into one or more crowdsourcing tasks, which are sufficiently simple to be carried out by non-experts and with a good quality. What are helpful here are reusable patterns for how best to crowdsource different kinds of NLP corpora. The GATE Crowdsourcing plugin

[1]Currently for CrowdFlower, which unlike Amazon Mechanical Turk is available globally.

currently provides such patterns for selection and classification tasks.

This stage also focuses on setup of the task parameters (e.g. number of crowd workers per task, payment per task) and piloting the project, in order to tune in its design. With respect to task parameters, infrastructural support is helpful, in order to enable automatic splitting of longer documents across crowdsourcing tasks.

2.2 Data Preparation

This stage, in particular, can benefit significantly from infrastructural support and reusable components, in order to collect the data (e.g. crawl the web, download samples from Twitter), preprocess it with linguistic tools (e.g. tokenisation, POS tagging, entity recognition), and then map automatically from documents and sentences to crowdsourcing micro-tasks.

2.3 Running the Crowdsourcing Project

This is the main phase of each crowdsourcing project. It consists of three kinds of tasks: task workflow and management, contributor management (including profiling and retention), and quality control. Paid-for marketplaces like Amazon Mechanical Turk and CrowdFlower already provide this support. As with conventional corpus annotation, quality control is particularly challenging, and additional NLP-specific infrastructural support can help.

2.4 Data Evaluation and Aggregation

In this phase, additional NLP-specific, infrastructural support is needed for evaluating and aggregating the multiple contributor inputs into a complete linguistic resource, and in assessing the resulting overall quality.

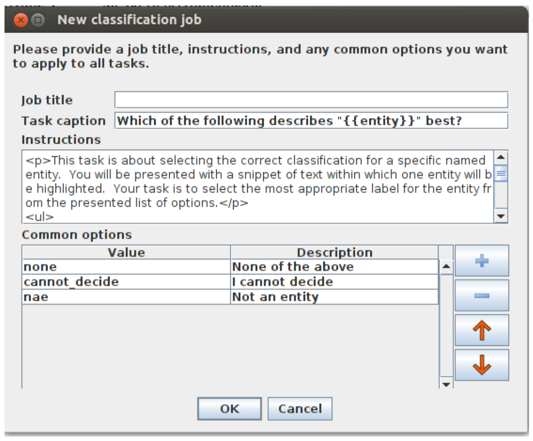
Next we demonstrate how these challenges have been addressed in our work.

3 The GATE Crowdsourcing Plugin

To address these NLP-specific requirements, we implemented a generic, open-source GATE Crowdsourcing plugin, which makes it very easy to set up and conduct crowdsourcing-based corpus annotation from within GATE's visual interface.

3.1 Physical representation for documents and annotations

Documents and their annotations are encoded in the GATE stand-off XML format (Cunningham


New classification job

Please provide a job title, instructions, and any common options you want to apply to all tasks.

Job title

Task caption: Which of the following describes "{{entity}}" best?

Instructions

```
<p>This task is about selecting the correct classification for a specific named
entity.  You will be presented with a snippet of text within which one entity will b
e highlighted.  Your task is to select the most appropriate label for the entity fr
om the presented list of options.</p>
<ul>
```

Common options

Value	Description
none	None of the above
cannot_decide	I cannot decide
nae	Not an entity

Figure 1: Classification UI Configuration

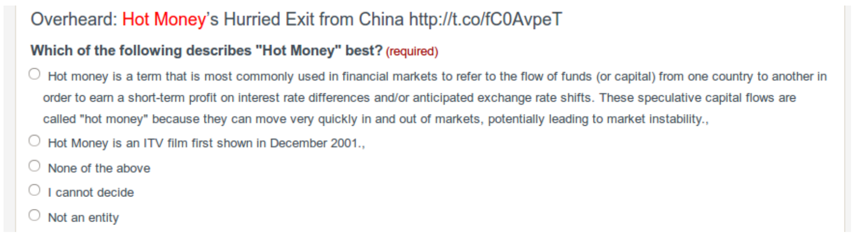
et al., 2002), which was chosen for its support for overlapping annotations and the wide range of automatic pre-processing tools available. GATE also has support for the XCES standard (Ide et al., 2000) and others (e.g. CoNLL) if preferred. Annotations are grouped in separate annotation sets: one for the automatically pre-annotated annotations, one for the crowdsourced judgements, and a consensus set, which can be considered as the final resulting corpus annotation layer. In this way, provenance is fully tracked, which makes it possible to experiment with methods that consider more than one answer as potentially correct.

3.2 Automatic data mapping to CrowdFlower

The plugin expects documents to be presegmented into paragraphs, sentences and word tokens, using a tokeniser, POS tagger, and sentence splitter – e.g. those built in to GATE (Cunningham et al., 2002). The GATE Crowdsourcing plugin allows choice between these of which to use as the crowdsourcing task unit; e.g., to show one sentence per unit or one paragraph. In the demonstration we will show both automatic mapping at sentence level (for named entity annotation) and at paragraph level (for named entity disambiguation).

3.3 Automatic user interface generation

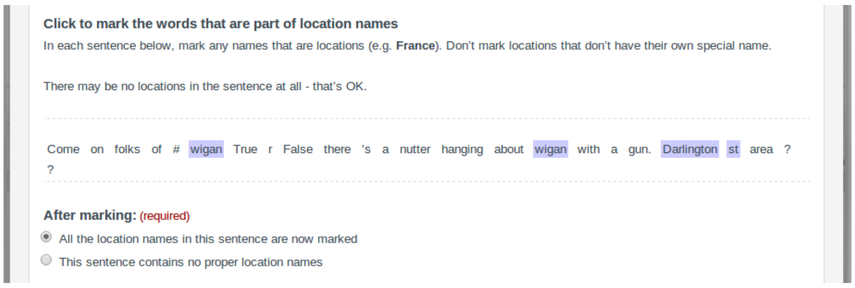
The User Interfaces (UIs) applicable to various task types tend to fall into a set of categories, the most commonly used being categorisation, selection, and text input. The GATE Crowdsourcing plugin provides generalised and re-usable, automatically generated interfaces for categorisation


Overheard: Hot Money's Hurried Exit from China http://t.co/fC0AvpeT

Which of the following describes "Hot Money" best? (required)

- ○ Hot money is a term that is most commonly used in financial markets to refer to the flow of funds (or capital) from one country to another in order to earn a short-term profit on interest rate differences and/or anticipated exchange rate shifts. These speculative capital flows are called "hot money" because they can move very quickly in and out of markets, potentially leading to market instability.,
- ○ Hot Money is an ITV film first shown in December 2001.,
- ○ None of the above
- ○ I cannot decide
- ○ Not an entity

Figure 2: Classification Interface: Sense Disambiguation Example


Click to mark the words that are part of location names

In each sentence below, mark any names that are locations (e.g. **France**). Don't mark locations that don't have their own special name.

There may be no locations in the sentence at all - that's OK.

Come on folks of # wigan True r False there 's a nutter hanging about wigan with a gun. Darlington st area ? ?

After marking: (required)

- ● All the location names in this sentence are now marked
- ○ This sentence contains no proper location names

Figure 3: Sequential Selection Interface: Named Entity Recognition Example

and selection.

In the first step, task name, instructions, and classification choices are provided, in a UI configuration dialog (see Figure 1). In this example, the instructions are for disambiguating named entities. We have configured three fixed choices, which apply to each entity classification task.

For some categorisation NLP annotation tasks (e.g. classifying sentiment in tweets into positive, negative, and neutral), fixed categories are sufficient. In others, where the available category choices depend on the text that is being classified (e.g. the possible disambiguations of Paris are different from those of London), choices are defined through annotations on each of the classification targets. In this case case, the UI generator then takes these annotations as a parameter and automatically creates the different category choices, specific to each crowdsourcing unit. Figure 2 shows an example for sense disambiguation, which combines two unit-specific classes with the three fixed classification categories shown before.

Figure 3 shows the CrowdFlower-based user interface for word-constrained sequential selection, which in this case is parameterised for named entity annotation. In sequential selection, sub-units are defined in the UI configuration – tokens, for this example. The annotators are instructed to click on all words that constitute the desired sequence (the annotation guidelines are given as a parameter during the automatic user interface generation).

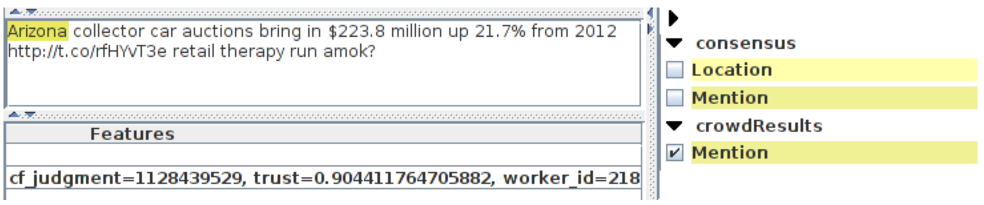
Since the text may not contain a sequence to be annotated, we also generate an explicit confirmation checkbox. This forces annotators to declare that they have made the selection or there is nothing to be selected in this text. CrowdFlower can then use gold units and test the correctness of the selections, even in cases where no sequences are selected in the text. In addition, requiring at least some worker interaction and decision-making in every task improves overall result quality.

3.4 Quality control

The key mechanism for spam prevention and quality control in CrowdFlower is test data, which we also refer to as gold units. These are completed examples which are mixed in with the unprocessed data shown to workers, and used to evaluate worker performance. The GATE Crowdsourcing plugin supports automatic creation of gold units from GATE annotations having a feature `correct`. The value of that feature is then taken to be the answer expected from the human annotator. Gold units need to be 10%–30% of the units to be annotated. The minimum performance threshold for workers can be set in the job configuration.

3.5 Automatic data import from CrowdFlower and adjudication

On completion, the plugin automatically imports collected multiple judgements back into GATE



Arizona collector car auctions bring in $223.8 million up 21.7% from 2012 http://t.co/rfHYvT3e retail therapy run amok?

Features

cf_judgment=1128439529, trust=0.904411764705882, worker_id=218

Figure 4: CrowdFlower Judgements in GATE

and the original documents are enriched with the crowdsourced information, modelled as multiple annotations (one per contributor). Figure 4 shows judgements that have been imported from CrowdFlower and stored as annotations on the original document. One useful feature is the trust metric, assigned by CrowdFlower for this judgement.

GATE's existing tools for calculating inter-annotator agreement and for corpus analysis are used to gain further insights into the quality of the collected information. If manual adjudication is required, GATE's existing annotations stack editor is used to show in parallel the annotations imported from CrowdFlower, so that differences in judgement can easily be seen and resolved. Alternatively, automatic adjudication via majority vote or other more sophisticated strategies can be implemented in GATE as components.

4 Conclusion

This paper described the GATE Crowdsourcing plugin[2] and the reusable components that it provides for automatic mapping of corpora to micro-tasks and vice versa, as well as the generic sequence selection and classification user interfaces. These are easily configurable for a wide range of NLP corpus annotation tasks and, as part of this demonstration, several example crowdsourcing projects will be shown.

Future work will focus on expanding the number of reusable components, the implementation of reusable automatic adjudication algorithms, and providing support for crowdsourcing through games-with-a-purpose (GWAPs).

Acknowledgments This was part of the uComp project (www.ucomp.eu). uComp receives the funding support of EPSRC EP/K017896/1, FWF 1097-N23, and ANR-12-CHRI-0003-03, in the framework of the CHIST-ERA ERA-NET.

[2]It is available to download from http://gate.ac.uk/ .

References

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus. Roberts, Valentin. Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. GATE Teamware: A Web-based, Collaborative Text Annotation Framework. *Language Resources and Evaluation*, 47:1007—1029.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: an Architecture for Development of Robust HLT Applications. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 7–12 July 2002*, ACL '02, pages 168–175, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karen Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Amazon mechanical turk: Gold mine or coal mine? *Computational Linguistics*, 37(2):413 –420.

Eduard Hovy. 2010. Annotation. In *Tutorial Abstracts of ACL*.

N. Ide, P. Bonhomme, and L. Romary. 2000. XCES: An XML-based Standard for Linguistic Corpora. In *Proceedings of the second International Conference on Language Resources and Evaluation (LREC 2000), 30 May – 2 Jun 2000*, pages 825–830, Athens, Greece.

Shashank Khanna, Aishwarya Ratan, James Davis, and William Thies. 2010. Evaluating and improving the usability of Mechanical Turk for low-income workers in India. In *Proceedings of the first ACM symposium on computing for development*. ACM.

Manfred Stede and Chu-Ren Huang. 2012. Interoperability and reusability: the science of annotation. *Language Resources and Evaluation*, 46:91–94. 10.1007/s10579-011-9164-x.

A. Wang, C.D.V. Hoang, and M. Y. Kan. 2012. Perspectives on Crowdsourcing Annotations for Natural Language Processing. *Language Resources and Evaluation*, Mar:1–23.

A Spinning Wheel for YARN: User Interface for a Crowdsourced Thesaurus

Pavel Braslavski
Ural Federal University
Kontur Labs
`pbras@yandex.ru`

Dmitry Ustalov
Ural Federal University
IMM UB RAS
`dau@imm.uran.ru`

Mikhail Mukhin
Ural Federal University
`mfly@sky.ru`

Abstract

YARN (Yet Another RussNet) project started in 2013 aims at creating a large open thesaurus for Russian using crowdsourcing. This paper describes synset assembly interface developed within the project — motivation behind it, design, usage scenarios, implementation details, and first experimental results.

1 Introduction

Creation of linguistic resources and annotations using crowdsourcing and gamification is becoming a common practice. Untrained workers contribute to development of thesauri, dictionaries, translation memories, and corpora annotations. Crowdsourcing can employ both paid workers, e.g. on Amazon Mechanical Turk (AMT) platform[1] and volunteers as in case of Wiktionary[2] — a large wiki-style online multilingual dictionary.

The goal of the YARN (Yet Another RussNet) project[3] launched in 2013 is to create a large open thesaurus for Russian language using crowdsourcing (Braslavski et al., 2013). Despite the fact that there were several attempts to create a Russian Wordnet (Azarova et al., 2002; Balkova et al., 2004; Gelfenbein et al., 2003; Loukachevitch and Dobrov, 2002), there is no open resource of acceptable quality and coverage currently available. The choice of crowdsourcing is also advocated by successful projects that are being evolved by volunteers: Russian Wiktionary[4], corpus annotation project OpenCorpora[5] and a wiki for linguistic resources related to Russian NLPub[6].

[1]`http://www.mturk.com/`
[2]`http://www.wiktionary.org/`
[3]`http://russianword.net/`
[4]`http://ru.wiktionary.org/`
[5]`http://opencorpora.org/`
[6]`http://nlpub.ru/`

Wordnets had been traditionally developed within small research teams. This approach maintains conceptual consistency and project manageability, facilitates informal exchange of ideas in a small group of contributors. However, this practice is hardly scalable and can potentially lead to a biased description of linguistic phenomena caused by the preferences of a close group of researchers. Crowdsourcing can possibly reduce costs, increase development pace, and make the results more robust, but puts additional demands on project management and tools, including user interface. Requirements for a crowdsourcing thesaurus development interface are as follows: 1) a low entry threshold for new users and a gradual learning curve; 2) no need for users to install additional software; 3) central data storage, collaborative work for several users in a competitive mode, and permission management; 4) change history tracking to protect data against vandalism.

Princeton WordNet editors had worked directly with lexicographer files stored in a version control system (Fellbaum, 1998). In later thesauri creation projects specialized tools were developed that featured more user-friendly interface, graphical representation of thesaurus relationships, centralized data storage, possibility of collaborative work, and data consistency checks. Examples of thesauri development tools are DEBVisDic (Horák et al., 2006), GernEdiT (Henrich and Hinrichs, 2010), as well as WordNetLoom (Piasecki et al., 2012) (see (Piasecki et al., 2012) for a brief overview of the-sauri editing tools). Wiktionary and OmegaWiki[7] use MediaWiki engine and wiki markup to encode dictionary information.

In the preparatory stage of the project we considered adoption of the above mentioned tools. However, we estimated that the amount of work needed for adaptation of existing tools to YARN

[7]`http://www.omegawiki.org/`

data formats and usage scenarios is quite costly and decided to develop a series of specialized tools.

The paper briefly describes YARN project and its noun synsets assembly interface in particular — motivation behind it, current state and appearance, usage scenarios, as well as results of a preliminary user study and future plans.

2 Project Outline

YARN is conceptually similar to Princeton Wordnet (Fellbaum, 1998) and its followers: it consists of synsets — groups of quasi-synonyms corresponding to a concept. Concepts are linked to each other, primarily via hierarchical hyponymic/hypernymic relationships. According to the project's outline, YARN contains nouns, verbs, and adjectives. We aim at splitting the process of thesaurus creation into smaller tasks and developing custom interfaces for each of them. The first step is an online tool for building noun synsets based on content of existing dictionaries. The goal of this stage is to establish YARN core content, test and validate crowdsourcing approach, prepare annotated data for automatic methods, and create a basis for the work with the other parts of speech.

As mentioned above, important characteristics of the project are its openness and recruitment of volunteers. Our crowdsourcing approach is different, for example, from the one described in (Biemann and Nygaard, 2010), where AMT turkers form synsets using the criterion of contextual substitutability directly. In our case, editors assemble synsets using word lists and definitions from dictionaries as "raw material". Obviously, such a task implies minimal lexicographical skills and is more complicated than an average task offered to AMT workers. Our target editors are college or university students, preferably from linguistics departments, who are native Russian speakers. It is desirable that students are instructed by a university teacher and may seek their advice in complex cases. As in the case of Wikipedia and Wiktionary, we foresee two levels of contributors: line editors and administrators with the corresponding privileges. According to our expectations, the total number of line editors can reach two hundreds throughout a year.

3 Raw Materials for YARN

We used two sources of "raw materials" for YARN: 1) Russian Wiktionary and 2) Small Academic Dictionary (SAD). Russian Wiktionary dump as of March 2012 was parsed and converted to database format using Wikokit software (Krizhanovsky and Smirnov, 2013). Wiktionary dump contains 51,028 nouns, including 45,646 single-word nouns; 30,031 entries have at least one definition. Besides the words and definitions Wiktionary dump contains occasionally synonym references and word usage examples. SAD data contain 33,220 word entries and 51,676 definitions. All single-word nouns were provided with frequencies based on the Russian National Corpus[8].

4 User Interface

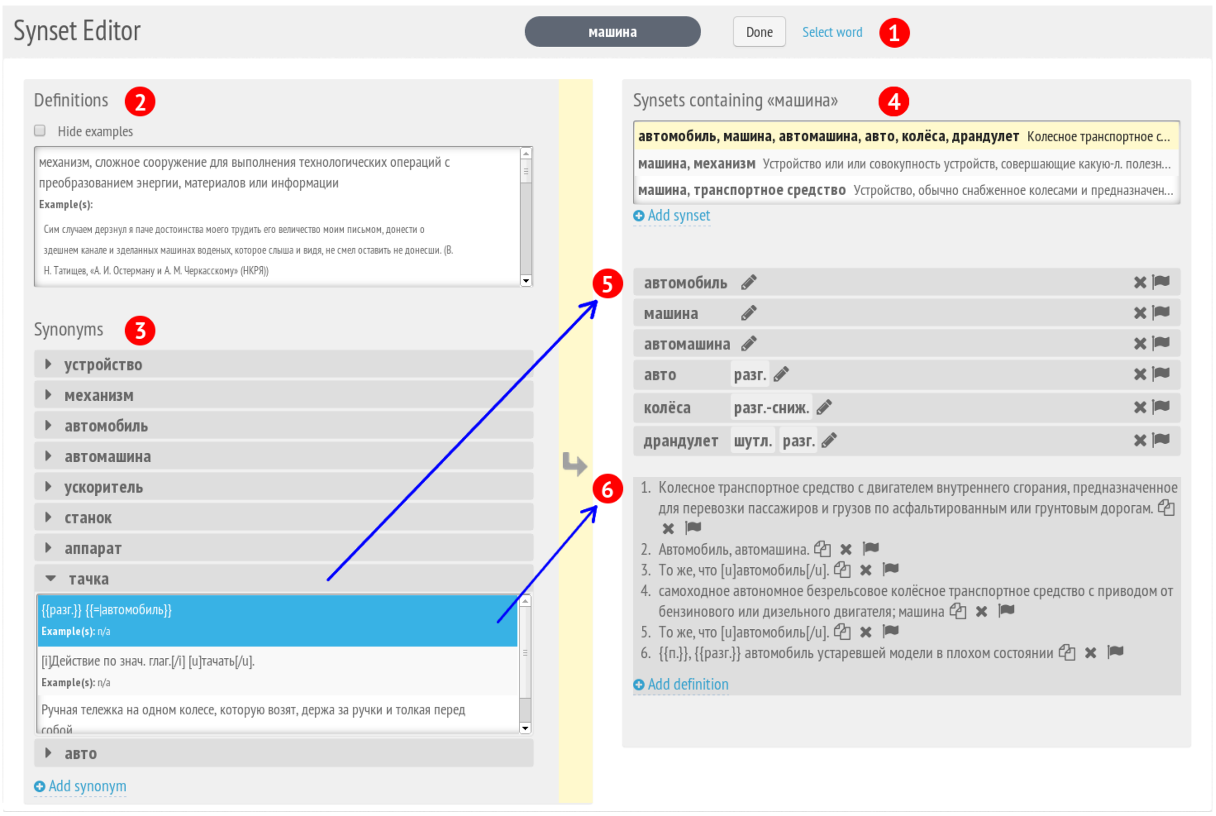
The current synset editing interface can be accessed online [9]; its main window is presented in Figure 1.

"Raw data" are placed on the left-hand side of the interface: definitions of the initial word and examples, possible synonyms for each of the meanings in turn with definitions and examples. The right-hand part represents resulted synsets including words, definitions, and examples. In principle, an editor can assemble a "minimal" synset from the dictionary "raw material" simply with several mouse clicks, without any typing.

Synset assembly begins with a word, or "synset starter". The editor selects an item from the list of words ranked by decreasing frequency; already processed words are shaded. The editor can go through the words one after another or choose an arbitrary word using search box.

The top left-hand pane displays definitions of the initial word and usage examples if any. To simplify the view, editor can turn out examples or to blind individual definitions. Possible synonyms of the initial word are listed at the bottom-left pane, in turn with definitions and examples. The top-right pane displays a list of synsets containing the initial word. The editor can copy definitions and usage examples of the initial word from the top left of the interface to the current synset with mouse clicks. From the synonyms pane one can transfer bare words or words along with definitions and examples. The editor can add a new word to the list

[8]`http://ruscorpora.ru/en/`
[9]`http://russianword.net/editor`



механизм, сложное сооружение для выполнения технологических операций с преобразованием энергии, материалов или информации

Example(s):

Сим случаем дерзнул я паче достоинства моего трудить его величество моим письмом, донести о здешнем канале и зделанных машинах воденых, которое слыша и видя, не смел оставить не донесши. (В. Н. Татищев, «А. И. Остерману и А. М. Черкасскому» (НКРЯ))

- устройство
- механизм
- автомобиль
- автомашина
- ускоритель
- станок
- аппарат
- тачка
  - {{разг.}} {{=|автомобиль}} **Example(s):** n/a
  - [i]Действие по знач. глаг.[/i] [u]тачать[/u]. **Example(s):** n/a
  - Ручная тележка на одном колесе, которую возят, держа за ручки и толкая перед собой
- авто

автомобиль, машина, автомашина, авто, колёса, драндулет Колесное транспортное с...
машина, механизм Устройство или или совокупность устройств, совершающие какую-л. полезн...
машина, транспортное средство Устройство, обычно снабженное колесами и предназначен...

автомобиль	
машина	
автомашина	
авто	разг.
колёса	разг.-сниж.
драндулет	шутл. разг.

1. Колесное транспортное средство с двигателем внутреннего сгорания, предназначенное для перевозки пассажиров и грузов по асфальтированным или грунтовым дорогам.
2. Автомобиль, автомашина.
3. То же, что [u]автомобиль[/u].
4. самоходное автономное безрельсовое колёсное транспортное средство с приводом от бензинового или дизельного двигателя; машина
5. То же, что [u]автомобиль[/u].
6. {{п.}}, {{разг.}} автомобиль устаревшей модели в плохом состоянии

Figure 1: Main window of YARN synset assembly interface (interface captions are translated for convenience of readers into English; originally all interface elements are in Russian): 1) initial word; 2) definitions and examples of the initial word; 3) possible synonyms of the initial word with definitions and examples; 4) a list of synsets containing the initial word (active synset is highlighted); 5) words constituting the current synset; 6) definitions of the current synset. The arrows show how the information items from the left-hand side form synsets in the right-hand side.

of synonyms; it will appear with dictionary definitions and examples if presented in the parsed data. If the editor is not satisfied with the collected definitions, they can create a new one — either from scratch or based on one of the existing descriptions. Additionally, a word or a definition within a synset can be flagged as "main"; and be provided with labels. All synset edits are tracked and stored in the database along with timestamps and editor ID.

YARN software is implemented using Ruby on Rails framework. All data are stored in a PostgreSQL database. User authentication is performed through an OAuth endpoint provided by Facebook. The user interface is implemented as a browser JavaScript application. The application interacts with the server application via JSON API. The entire source code of the project is available in an open repository[10].

[10]https://github.com/russianwordnet

5 Preliminary Results

In the fall 2013 we conducted a pilot user study with 45 students of the linguistics department at the Ural Federal University. The experiment resulted in 1390 synsets; 970 of them are 'non-trivial', i.e. contain more than a single word (253 contain 2 words, 228 — 3 words, 207 — 4, 282 — 5+). Editors spent about two minutes on building a 'non-trivial' synset on average, which we find a very good result. Figure 2 shows the distribution of edit times for 2+ word synsets. Distribution of completed synsets by students is also skewed, e.g. top-5 contributors account for more than a third of all non-trivial synsets (329).

Figure 3 shows a linear trend of time spent by five top contributors on constructing consecutive non-trivial synsets. Four out of five demonstrate a learning effect: average time per synset tends to decrease while the editor proceeds through tasks.

In general, students were very positive about

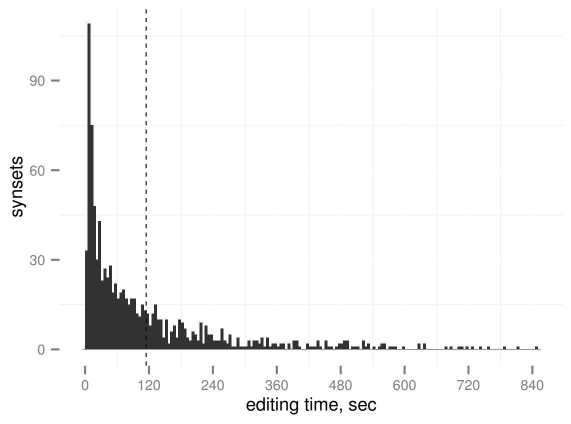


Figure 2: Distribution of times spent on non-trivial synset editing.

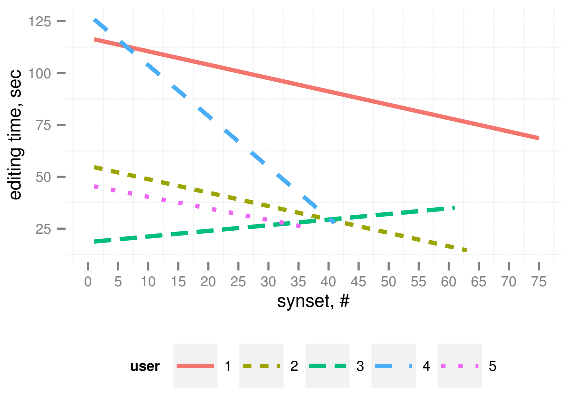


Figure 3: Linear trend of time spent on sequential edits of nontrivial synsets by top-5 contributors.

their participation in the experiment and the YARN interface. Participants mentioned flaws in parsed data, inability to delete an existing synset (we disabled this option during the experiment), and the inconvenience of label assignments as main disadvantages.

6 Conclusions

YARN synset assembly tool passed an initial testing and proved to be a usable tool for creation of thesaurus building blocks. Upon reading simple instructions, volunteers were able to quickly learn an intuitive interface and accomplish the synset assembly task without problems.

During the experiment we were able to diagnose some flaws related to interface design, editor guidelines, and internal data representation. In the future we will elaborate instructions and learning materials, clean existing and add more dictionary data, and perform a thorough evaluation of the interface. Then, we will work on an interface for linking synsets and expand YARN with verbs and adjectives.

Acknowledgments. This work is supported by the Russian Foundation for the Humanities, project #13-04-12020 “New Open Electronic Thesaurus for Russian”.

References

Irina Azarova et al. 2002. RussNet: Building a Lexical Database for the Russian Language. In *Proc. of Workshop on WordNet Structures and Standardisation, and How These Affect WordNet Applications and Evaluation, Gran Canaria, Spain*, pages 60–64.

Valentina Balkova et al. 2004. Russian wordnet. In *Proc. of the Second Global WordNet Conference*, pages 31–38. Citeseer.

Chris Biemann and Valerie Nygaard. 2010. Crowdsourcing Wordnet. In *Proc. of the 5th Global WordNet conference, Mumbai, India*.

Pavel Braslavski et al. 2013. YARN Begins. In *Proc. of Dialog-2013 (in Russian)*.

Christiane Fellbaum. 1998. WordNet: An Electronic Database.

Ilya Gelfenbein et al. 2003. Avtomaticheskij perevod semanticheskoj seti WORDNET na russkij yazyk. In *Proc. of Dialog'2003 (in Russian)*.

Verena Henrich and Erhard Hinrichs. 2010. GernEdiT-The GermaNet Editing Tool. In *ACL (System Demonstrations)*, pages 19–24.

Aleš Horák et al. 2006. DEBVisDic–First Version of New Client-Server Wordnet Browsing and Editing Tool. In *Proc. of the Third International Wordnet Conference*.

Andrew Krizhanovsky and Alexander Smirnov. 2013. An Approach to Automated Construction of a General Purpose Lexical Ontology Based on Wiktionary. *Journal of Computer and Systems Sciences International*, 52(2):215–225.

Natalia Loukachevitch and Boris Dobrov. 2002. Development and Use of Thesaurus of Russian Language RuThes. In *Proc. of Workshop on WordNet Structures and Standardisation, and How These Affect WordNet Applications and Evaluation, Gran Canaria, Spain*, pages 65–70.

Maciej Piasecki et al. 2012. WordnetLoom: a Wordnet Development System Integrating Form-based and Graph-based Perspectives. *International Journal of Data Mining, Modelling and Management*.

Author Index