9th SaLTMiL Workshop on

"Free/open-Source Language Resources for the Machine Translation of Less-Resourced Languages"

LREC 2014, Reykjavík, Iceland, 27 May 2014

Workshop Programme

09:00 – 09:30 Welcoming address by Workshop co-chair Mikel L. Forcada

09:30 – 10:30 Oral papers

Iñaki Alegria, Unai Cabezon, Unai Fernandez de Betoño, Gorka Labaka, Aingeru Mayor, Kepa Sarasola and Arkaitz Zubiaga
Wikipedia and Machine Translation: killing two birds with one stone

Gideon Kotzé and Friedel Wolff
Experiments with syllable-based English-Zulu alignment

10:30 – 11:00 Coffee break

11:00 – 13:00 Oral papers

Inari Listenmaa and Kaarel Kaljurand
Computational Estonian Grammar in Grammatical Framework

Matthew Marting and Kevin Unhammer
FST Trimming: Ending Dictionary Redundancy in Apertium

Hrvoje Peradin, Filip Petkovski and Francis Tyers
Shallow-transfer rule-based machine translation for the Western group of South Slavic languages

Alex Rudnick, Annette Rios Gonzales and Michael Gasser
Enhancing a Rule-Based MT System with Cross-Lingual WSD

13:00 – 13:30 General discussion

13:30 Closing

Editors

Mikel L. Forcada	Universitat d'Alacant, Spain
Kepa Sarasola	Euskal Herriko Unibertsitatea, Spain
Francis M. Tyers	UiT Norgga árktalaš universitehta, Norway

Workshop Organizers/Organizing Committee

Mikel L. Forcada	Universitat d'Alacant, Spain
Kepa Sarasola	Euskal Herriko Unibertsitatea, Spain
Francis M. Tyers	UiT Norgga árktalaš universitehta, Norway

Workshop Programme Committee

Iñaki Alegria	Euskal Herriko Unibertsitatea, Spain
Lars Borin	Göteborgs Universitet, Sweden
Elaine Uí Dhonnchadha	Trinity College Dublin, Ireland
Mikel L. Forcada	Universitat d'Alacant, Spain
Michael Gasser	Indiana University, USA
Måns Huldén	Helsingin Yliopisto, Finland
Krister Lindén	Helsingin Yliopisto, Finland
Nikola Ljubešić	Sveučilište u Zagrebu, Croatia
Lluís Padró	Universitat Politècnica de Catalunya, Spain
Juan Antonio Pérez-Ortiz	Universitat d'Alacant, Spain
Felipe Sánchez-Martínez	Universitat d'Alacant, Spain
Kepa Sarasola,	Euskal Herriko Unibertsitatea, Spain
Kevin P. Scannell	Saint Louis University, USA
Antonio Toral	Dublin City University, Ireland
Trond Trosterud	UiT Norgga árktalaš universitehta, Norway
Francis M. Tyers	UiT Norgga árktalaš universitehta, Norway

Table of contents

Author Index

Introduction

The 9th International Workshop of the Special Interest Group on Speech and Language Technology for Minority Languages (SaLTMiL) will be held in Reykjavík, Iceland, on 27th May 2014, as part of the 2014 International Language Resources and Evaluation Conference (LREC). (For SALTMIL see: http://ixa2.si.ehu.es/saltmil/); it is also framed as one of the activities of European project Abu-Matran (http://www.abumatran.eu). Entitled "Free/open-source language resources for the machine translation of less-resourced languages", the workshop is intended to continue the series of SALTMIL/LREC workshops on computational language resources for minority languages, held in Granada (1998), Athens (2000), Las Palmas de Gran Canaria (2002), Lisbon (2004), Genoa (2006), Marrakech (2008), La Valetta (2010) and Istanbul (2012), and is also expected to attract the audience of Free Rule-Based Machine Translation workshops (2009, 2011, 2012).

The workshop aims to share information on language resources, tools and best practice, to save isolated researchers from starting from scratch when building machine translation for a less-resourced language. An important aspect will be the strengthening of the free/open-source language resources community, which can minimize duplication of effort and optimize development and adoption, in line with the LREC 2014 hot topic ‘LRs in the Collaborative Age’ (http://is.gd/LREChot).

Papers describe research and development in the following areas:

- Free/open-source language resources for rule-based machine translation (dictionaries, rule sets)
- Free/open-source language resources for statistical machine translation (corpora)
- Free/open-source tools to annotate, clean, preprocess, convert, etc. language resources for machine translation
- Machine translation as a tool for creating or enriching free/open-source language resources for less-resourced languages

Wikipedia and Machine Translation: killing two birds with one stone

Iñaki Alegria (1), Unai Cabezon (1) , Unai Fernandez de Betoño (2), Gorka Labaka (1), Aingeru Mayor (1), Kepa Sarasola (1) and Arkaitz Zubiaga (3)

(1) Ixa Group, University of the Basque Country UPV/EHU,
(2)Basque Wikipedia and University of the Basque Country,
(3)Basque Wikipedia and Applied Intelligence Research Centre of the Dublin Institute of Technology

Informatika Fakultatea, Manuel de Lardizabal 1, 20013 Donostia (Basque Country)

E-mail: i.alegria@ehu.es

Abstract

In this paper we present the free/open-source language resources for machine translation created in OpenMT-2 wikiproject, a collaboration framework that was tested with editors of Basque Wikipedia. Post-editing of Computer Science articles has been used to improve the output of a Spanish to Basque MT system called Matxin. For the collaboration between editors and researchers, we selected a set of 100 articles from the Spanish Wikipedia. These articles would then be used as the source texts to be translated into Basque using the MT engine. A group of volunteers from Basque Wikipedia reviewed and corrected the raw MT translations. This collaboration ultimately produced two main benefits: (i) the change logs that would potentially help improve the MT engine by using an automated statistical post-editing system, and (ii) the growth of Basque Wikipedia. The results show that this process can improve the accuracy of a Rule Based Machine Translation system in nearly 10% benefiting from the post-edition of 50,000 words in the Computer Science domain. We believe that our conclusions can be extended to MT engines involving other less-resourced languages lacking large parallel corpora or frequently updated lexical knowledge, as well as to other domains.

Keywords: collaborative work, Machine Translation, Wikipedia, Statistical Post-Edition

1. Introduction

A way for improving Rule Based Machine Translation (RBMT) systems is to use a Statistical Post-Editor (SPE) that automatically post-edits the output of the MT engines. But building a SPE requires a corpus of MT outputs and their manual post-editions pairs.

We argue that creatively combining machine translation and human editing can benefit both article generation on Wikipedia, and the development of accurate machine translation systems.

One of the key features on the success of Wikipedia, the popular and open online encyclopaedia, is that it is available in more than 200 languages. This enables the availability of a large set of articles in different languages. The effort of Wikipedia editors to keep contents updated, however, increases as the language has a smaller community of editors. Because of this, less-resourced languages with smaller number of editors cannot keep pace with the rapid growth of top languages such as English Wikipedia. To reduce the impact of this, editors of small Wikipedias can take advantage of contents produced in top languages, so they can generate large amounts of information by translating those. To relax such process of translating large amounts of information, machine translation provides a partially automated solution to potentially facilitate article generation (Way, 2010). This presents the issue that current machine translation systems generate inaccurate translations that require substantial post-editing by human editors.

In this paper, we introduce our methodology to enable collaboration between Wikipedia editors and researchers, as well as the system we have developed accordingly. This system permits the generation of new articles by editing machine translation outputs, while editors help improve a machine translation system. We believe that amateur translators can benefit from MT rather than professional translators.

Specifically, to perform such collaboration between editors and researchers, a set of 100 articles were selected from Spanish Wikipedia to be translated into Basque using the machine translation (MT) system called Matxin (Mayor et al., 2011). A group of volunteers from Basque Wikipedia reviewed and corrected these raw translations. In the correction process, they could either post-edit the MT output to fix errors, or retranslate it when the machine-provided translation was inaccurate. We logged their changes, and stored the final article generated. This process ultimately produced two main benefits: (i) a set of free/open-source language resources for machine translation, among others the change logs that potentially help improve the MT engine

by using an automated statistical post-editor (Simard et al., 2007), and (ii) the generated articles that expand the Basque Wikipedia. The results show that this process can improve the accuracy of an Rule Based MT (RBMT) system in nearly 10% benefiting from the post-edition of 50,000 words in the Computer Science domain (Alegria et al., 2013). This improvement was

Section 2 defines the methodology followed in this collaborative project: its design, the criteria and tools used to select the set of Wikipedia articles to be translated, and the resources used to adapt the general MT system to the domain of computer science. Then Section 3 presents the free/open-source language resources and tools created in this project and the achieved translation improvements. The paper ends with the conclusions and future work.

2. Related work

Statistical post-editing (SPE) is the process of training a Statistical Machine Translation (SMT) system to translate from rule- based MT (RBMT) outputs into their manually post-edited versions (Simard et al., 2007). They report a reduction in post-editing effort of up to a third when compared to the output of the RBMT system. Isabelle et al. (2007) later confirmed those improvements. A corpus with 100,000 words of post- edited translations outperformed a lexicon-enriched baseline RBMT system.

Using SPE and SYSTRAN as the RBMT system, Dugast et al. (2007, and 2009) significantly improve the lexical choice of the final output. Lagarda et al. (2009) presented an average improvement of 59.5% in a real translation scenario that uses Euoperl corpus, and less significant improvements (6.5 %) when using a more complex corpus.

The first experiments performed for Basque were different because morphological modules were used in both RBMT and SMT translations, and because the size of available corpora was small (Díaz de Ilarraza and al., 2010). The post-edition corpus was artificially created from bilingual corpus: new RBMT translations for the source sentences and taking their target sentences in the bilingual corpus as the post-edited sentences. The improvements they report when using an RBMT+SPE approach on a restricted domain are bigger than when using more general corpora.

Some frameworks for collaborative translation have been created recently. (1) Cross-Lingual Wiki Engine was presented in 2008 (Huberdeau, et al. , 2008). (2) In 2011, the company Asia Online translated 3.5 million articles from English Wikipedia into Thai using MT. (3) Users registered in Yeeyan.org collaboratively translate Wikipedia articles from English to Chinese. (4) Wasala et al. (2013) created a client-server architecture, used in Web localization, to share and use translation memories, which can be used to build (or improve) MT systems. (5) And 'Collaborative Machine Translation for Wikipedia' 1is a Wikimedia proposal for a long-term strategy using several technologies for offering a machine translation system based on collaborative principles. (6) an experiment focused on post-editon of MT output of wiki entries from German and Dutch into English (Gaspari et al., 2011) report that overall the users were satisfied with the system and regarded it as a potentially useful tool to support their work; in particular, they found that the post-editing effort required to attain translated wiki entries in English of publishable quality was lower than translating from scratch.

Popular MT engines include a post-edition interface to fix translations. For instance, Google Translate[2] allows its users to post-edit translations by replacing or reordering words. These corrections, which are only internally available to Google, provide valuable knowledge to enhance the system for future translations. Other companies such as Lingotek,[3] sell Collaborative Translation Platforms that include post-edition capabilities. For our collaborative work, we use OmegaT, an open source Computer Aided Translation (CAT) tool.

3. Design and methodology of the collaborative project on translation

This collaboration among computer scientists, linguists and editors of Basque Wikipedia was developed within the OpenMT-2 Wikiproject. The objective was to design and develop a final MT system by building a Statistical Post-Editor that automatically post-edits the output of the original RBMT system.

To perform such collaboration between editors and researchers, a set of 100 articles from Spanish Wikipedia were translated into Basque using the Matxin RBMT engine. A group of volunteers reviewed and corrected these raw translations. In the correction process, they could either post-edit the MT output to fix errors, or retranslate it when the machine-provided translation was inaccurate. With the aim of facilitating the post-edition task for editors, we adapted the well- known open-source tool OmegaT.

To improve the quality of the Matxin RBMT system's outputs given to the post-editors, we adapted Matxin to the Computer Science domain and the Wikipedia articles to be translated in the project were selected from the Computer Science category. We choose this domain, both because it is suitable as a domain that does not highly depend on cultural factors and because it is a well known domain for our research group.

---

[1] https://meta.wikimedia.org/wiki/Collaborative_Machine_Translation_for_Wikipedia

[2] http://translate.google.com

[3] http://lingotek.com

The public collaboration campaign was run for eight months, from July 2011 to February 2012 and 36 volunteers collaborated in it. This process ultimately produced two main benefits:

1. The raw and manual post-edited translation pairs served to built an automated Statistical Post-Editor. This SPE system can improve the accuracy of the RBMT system in nearly 10%. MBLEU, BLEU, NIST, METEOR, TER,WER and PER metrics confirm this improvement (Alegria et al, 2013).
2. The generated articles help expand the Basque Wikipedia. 100 new entries (50,204 words) had been added to the Basque Wikipedia.

Additionally, improvements have been made in both Matxin and OmegaT systems.

3.1 Selection of Wikipedia articles

To incorporate new collaborators that are sometimes not very motivated to participate in work excessively long we decided to translate short Wikipedia articles.

We created a tool to help us search for short untranslated Wikipedia entries. This tool is a perl script named wikigaiak4koa.pl that, given a Wikipedia category and four languages, returns the list of articles contained in the category with their corresponding equivalents in those four languages and their length.

The size of the Catalan Wikipedia (378,408 articles) is midway between the Spanish (902,113 articles) and the Basque (135,273 articles). Therefore, we consider that a Wikipedia article that is present in the Catalan Wikipedia but not in the Basque Wikipedia should be included in the latter before other non-existing articles that are not in the Catalan version.

Using the tool we identified 140 entries that: (1) were included in the Catalan and Spanish Wikipedias, (2) were not in the Basque Wikipedia, and (3) the size in the Spanish Wikipedia was smaller than 30 Kb (~ 30,000 characters). These 140 intermediate size entries were included in the Wikiproject.

The script can be used to examine the contents of any Wikipedia category for any language.

3.2 Modifications to Matxin RBMT system

The Basque-Spanish Matxin RBMT system was adapted to the Computer Science domain. The bilingual lexicon was customized in two ways:

- Adaptation of lexical resources from dictionary-systems. Using several Spanish/Basque on-line dictionaries, we performed a systematic search for word meanings in the Computer Science domain. We included 1,623 new entries in the lexicon of the original RBMT system. The new terms were mostly multi-words, such as *base de datos* (database) and *lenguaje de programación* (programming language). Some new single words were also obtained; for example, *iterativo* (iterative), *ejecutable* (executable) or *ensamblador* (assembly). In addition, the lexical selection was changed for 184 words: e.g. *rutina-ERRUTINA* (routine) before *rutina-OHITURA* (habit).
- Adaptation of the lexicon from a parallel corpus. We collected a parallel corpus in the Computer Science domain from the localized versions of free software from Mozilla, including Firefox and Thunderbird (138,000 segments, 600,000 words in Spanish and 440,000 in Basque). We collected the English/Basque and the English/Spanish localization versions and then generated a new

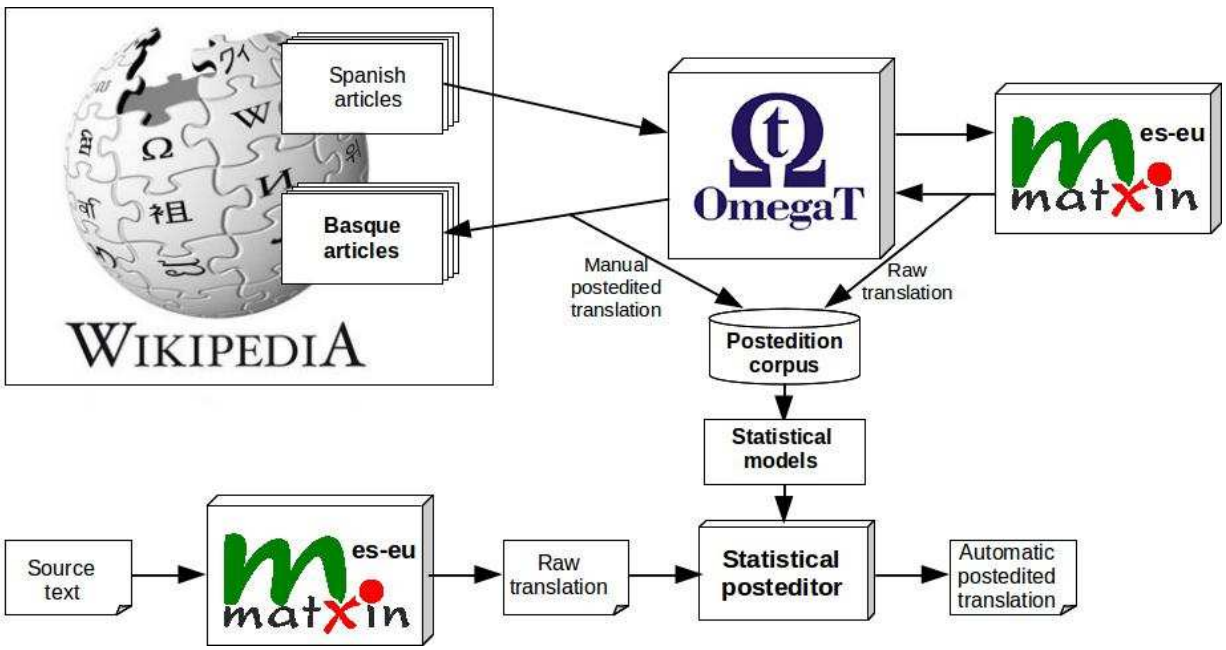

Figure 1. Architecture of the final MT system enriched with a Statistical Posteditor.

parallel corpus for the Spanish/Basque language pair, now publicly available. These texts may not be suitable for SMT but they are useful for extracting lexical relations. Based on Giza+alignments, we extracted the list of possible translations as well as the probability of each particular translation for each entry in the corpus. In favour of precision, we limited the use of these lists to the lexical selection. The order was modified in 444 dictionary entries. For example, for the Spanish term *dirección*, the translated word *HELBIDE* (address) was selected instead of *NORABIDE* (direction).

3.3 Modifications to OmegaT

OmegaT was selected as the post-edition platform to be used in our project. To make it easier to use for editors, we adapted the interface of OmegaT with a number of additional features:

- Integration of Matxin Spanish to Basque MT engine. OmegaT includes a class that connects several machine translation services, making it relatively easy to customize by adding more services. We used this class to integrate Matxin within OmegaT. In order to reduce the integration effort, we made Matxin's code simpler, lighter and more readable so that it could be implemented as a web service to be accessed by single API calls using SOAP. Therefore, OmegaT could easily make use of a Spanish to Basque machine translation system.
- Integration of the Basque speller, to facilitate post-editing.
- A functionality to import/export of Wikipedia articles to/from OmegaT. We implemented a new feature to upload the translated article to the Basque Wikipedia to OmegaT's existing capability of importing MediaWiki documents from their URL encoded as UTF8. To enable this new feature, we also implemented a new login module and some more details. When uploading an article to Wikipedia, the editor is also required to provide a copy of the translation memory created with the article. We use these translation memories in the process of building the SPE system.
- A tool for translating Wikipedia links. This module use Wikipedia metadata to search the Basque article that corresponds to a Spanish one. As an example of translation of Wikipedia metadata, let us take the translation of the internal Wikipedia link [[*gravedad* | *gravedad*]] in the Spanish Wikipedia (equivalent to the link [[gravity | gravity]] in the English Wikipedia). Our system translates it as [[*GRABITAZIO* | *LARRITASUNA*]], so it translates the same word in a different way when it represents the entry Wikipedia and when it is the text shown in such a link. On the one hand, the link to the entry *gravedad* in the Spanish Wikipedia is translated as *GRABITAZIO* (gravitation) making use of the mechanics of MediaWiki documents which include information on the languages in which a particular entry is available, and their corresponding entries. And on the other hand, the text word *gravedad* is translated as *LARRITASUNA* (seriousness) using the RBMT system. Therefore, this method provides a translation adapted to Wikipedia. Offering this option allows the post-editor to correct the RBMT translation with the usually more suitable "Wikipedia translation".

4. Created resources and achieved improvements

The complete set of publicly available resources created in this project includes the following products:

- Corpus
  - The new Spanish/Basque version of the parallel corpus[4]. created from the localized versions of free software from Mozilla (138.000 segments, 600.000 word in Spanish and 440.000 in Basque).
  - The corpus[5] of raw and manual post-edited translations (50.204 words). It was created by manual post-editing of the Basque outputs given by Matxin RBMT system translating 100 entries from the Spanish Wikipedia.
- Wikipedia
  - The 100 new entries[6] added to Basque Wikipedia (50.204 words).
  - A tool for searching articles in the Wikipedia (wikigaiak4koa.pl [7]). This tool is a perl script that can be used to browse the content of a category for any language in Wikipedia. Given a Wikipedia category and four languages, it returns the list of articles contained in the category with their corresponding equivalents in those four languages and their length.
- Matxin

4 http://ixa2.si.ehu.es/glabaka/lokalizazioa.tmx
5 http://ixa2.si.ehu.es/glabaka/OmegaT/OpenMT-OmegaT-CS-TM.zip
6 http://eu.wikipedia.org/w/index.php?title=Berezi:ZerkLotzenDuHona/Txantiloi:OpenMT-2&limit=250
7 http://www.unibertsitatea.net/blogak/testuak-lantzen/2011/11/22/wikigaiak4koa

- The new version of the Matxin RBMT system customized for the domain of Computer Science available as a SOAP service.[8]
- A new automated Statistical Post-Editing system. This system has been built using the corpus of raw RBMT translation outputs and their corresponding manual post-editions (50.204 words).
- The quantitative results show that the combination of RBMT-SPE pipeline can improve the accuracy of the raw RBMT system at around 10%, despite the fact that the size of the corpus used to built the SPE system is smaller than those referenced in the major contributions to SPE (for example, Simard et al. used a corpus of 100,000 words). Thus, there may be room for further improvement by the simple expedient of using a larger post-edition corpus.

- OmegaT
  - Integration of Matxin Spanish to Basque MT engine.
  - Integration of the Basque speller.
  - A functionality to import/export of Wikipedia articles to/from OmegaT. This upload is language-independent, and can be used for languages other than Basque. However, this feature has not been tested yet on languages that rely on different character sets such as CJK or Arabic.
  - A tool for translating Wikipedia links. This module use Wikipedia metadata to search the Basque article that corresponds to a Spanish one.
  - A tutorial in Basque to download, install and use OmegaT, with details to post-edit Wikipedia articles[9].

5. Conclusions and Future Work

Creating and coordinating a community to produce materials for a less resourced language can be a substantial task. We have defined a collaboration framework that enables Wikipedia editors to generate new articles while they help development of machine translation systems by providing post-edition logs. This collaboration framework has been experimented with editors of Basque Wikipedia. Their post-editing on Computer Science articles were used to train a SPE system that improves the output of the Spanish to Basque MT system called Matxin.

We set forth the hypothesis that MT could be helpful to amateur translators even if not so much to professionals. We can confirm our hypothesis, as even when the quality of the MT output was not high, it was enough to prove useful in helping the editors perform their work. We also observed that Wikipedia metadata makes more complicated both the MT and the post-editing processes, even if the use of Wikipedia's interlanguage links effectively help translation.

The benefits of this project were twofold: improvement of the outputs of the MT system, and extension the Basque Wikipedia with new articles. Various auxiliary tools and language resources developed as part of this research can also be considered as valuable resources for other collaborative projects.

6. References

Alegria I., Cabezon U., Fernandez de Betoño U., Labaka G., Mayor A., Sarasola K., Zubiaga A. (2013) Reciprocal Enrichment between Basque Wikipedia and Machine Translators. In I. Gurevych and J. Kim (Eds.) The People's Web Meets NLP: Collaboratively Constructed Language Resources', Springer. ISBN-10: 3642350844, pp. 101-118.

Diaz de Ilarraza A., Labaka G., Sarasola K. (2008) Statistical post-editing: a valuable method in domain adaptation of RBMT systems. In: Proceedings of MATMT2008 workshop: mixing approaches to machine translation, Euskal Herriko Unibersitatea, Donostia, pp 35–40

Dugast L., Senellart J., Koehn P. (2007) Statistical post-editing on SYSTRAN's rule-based translation system. In: Proceedings of the second workshop on statistical machine translation, Prague, pp 220–223

Dugast L., Senellart J., Koehn P. (2009) Statistical post editing and dictionary extraction: Systran/Edinburgh submissions for ACL-WMT2009. In: Proceedings of the fourth workshop on statistical machine translation, Athens, pp 110–114

Gaspari F., Toral A., and Naskar S. K. (2011) User-focused Task-oriented MT Evaluation for Wikis: A Case Study.. In *Proceedings of the Third Joint EM+/CNGL Workshop "Bringing MT to the User:* Research *Meets Translators"*. European Commission, Luxembourg, 14 October 2011. 13-22

Huberdeau L.F., Paquet B., Desilets A. (2008). The Cross-Lingual Wiki Engine: enabling collaboration across language barriers. In *Proceedings of the 4th International Symposium on Wikis* (WikiSym '08).

8 htt p : //ixa2.si.ehu.es/matxin erb/translate.cgi
9 http://siuc01.si.ehu.es/~jipsagak/OpenMT_Wiki/Eskuliburua_Euwikipedia+Omegat+Matxin.pdf

ACM, New York, NY, USA

Isabelle P., Goutte C., Simard M. (2007) Domain adaptation of MT systems through automatic post-editing. In: Proceedings of the MT Summit XI, Copenhagen, pp 255–261

Lagarda A.L., Alabau V., Casacuberta F., Silva R., Díaz-de-Liaño E. (2009) Statistical post-editing of a rule-based machine translation system. In: Proceedings of NAACL HLT 2009. Human language technologies: the 2009 annual conference of the North American chapter of the ACL, Short Papers, Boulder, pp 217–220

Mayor A., Diaz de Ilarraza A., Labaka G., Lersundi M., Sarasola K. (2011) Matxin, an open-source rule-based machine translation system for Basque. Machine Translation Journal 25(1):53–82

Simard M., Ueffing N., Isabelle P., Kuhn R. (2007) Rule-based translation with statistical phrase- based post-editing. In: Proceedings of the second workshop on statistical machine translation, Prague, pp 203–206

Wasala A., Schäler R., Buckley J., Weerasinghe R., Exton C. (2013) Building Multilingual Language Resources in Web Localisation: A Crowdsourcing Approach.. In I. Gurevych and J. Kim (Eds.) The People's Web Meets NLP: Collaboratively Constructed Language Resources', Springer. ISBN-10: 3642350844, pp. 69-100.

Way A. (2010) Machine translation. In: Clark A, Fox C, Lappin S (eds) The handbook . of computational linguistics and natural language processing. Wiley-Blackwell, Oxford, pp 531–573

Experiments with syllable-based English-Zulu alignment

Gideon Kotzé, Friedel Wolff

University of South Africa
kotzegj@unisa.ac.za, wolfff@unisa.ac.za

Abstract

As a morphologically complex language, Zulu has notable challenges aligning with English. One of the biggest concerns for statistical machine translation is the fact that the morphological complexity leads to a large number of words for which there exist very few examples in a corpus. To address the problem, we set about establishing an experimental baseline for lexical alignment by naively dividing the Zulu text into syllables, resembling its morphemes. A small quantitative as well as a more thorough qualitative evaluation suggests that our approach has merit, although certain issues remain. Although we have not yet determined the effect of this approach on machine translation, our first experiments suggest that an aligned parallel corpus with reasonable alignment accuracy can be created for a language pair, one of which is under-resourced, in as little as a few days. Furthermore, since very little language-specific knowledge was required for this task, our approach can almost certainly be applied to other language pairs and perhaps for other tasks as well.
Keywords: machine translation, morphology, alignment

1. Introduction

Zulu is an agglutinative language in the Bantu language family. It is written in a conjunctive way which results in words that can contain several morphemes. Verbs are especially prone to complex surface forms. Although word alignment algorithms might have enough information to align all the words in an English text to their Zulu counterparts, the resulting alignment is not very useful for tasks such as machine translation because of the sparseness of morphologically complex words, even in very large texts. This is compounded by the fact that Zulu is a resource-scarce language.

A possible solution for this problem is to morphologically analyze each word and using the resulting analysis to split it into its constituent morphemes. This enables a more fine-grained alignment with better constituent convergence. Since verb prefixes often denote concepts such as subject, object, tense and negation, it would be ideal if they would align with their (lexical) counterparts in English. Figure 1 shows an example of a Zulu-English alignment before and after the segmentation. Here, it is clear that not only more alignments can be made, but in some cases, such as with *of*, we have better convergence as well.

Figure 1: An example of an English-Zulu alignment before and after morphological segmentation.

Variations of this strategy have been followed with some success with similar language pairs such as English-Swahili (De Pauw et al., 2011) and English-Turkish (Çakmak et al., 2012).[1]

Morphological analysers are, however, difficult and time consuming to develop, and often relatively language specific. Although the bootstrapping of morphological analysers between related languages shows promise (Pretorius and Bosch, 2009), in each case the construction of a language-specific lexicon is still required, which is a large amount of work. The Bantu language family is considered resource-scarce, and methods that rely on technologies such as morphological analyzers, will mostly be out of reach for languages in this family.

We approach this problem by noting the fact that most languages in the Bantu family have a preference for open syllables (Spinner, 2011) and that in our case, even a simple syllabification approach can roughly approximate morphological segmentation. Hyman (2003) states that the open syllable structure of Proto-Bantu is reinforced by the agglutinative morphology. It is therefore possible to decompose words accurately for many Bantu languages into syllables in a straightforward way. If syllabification is useful for the task of word alignment (or indeed, any other task), it could be applicable to a large number of under-resourced languages. Indeed, some success has been demonstrated by Kettunen (2010) for an information retrieval task in several European languages. As far as we are aware, a syllable-based approach to alignment has not yet been implemented for Bantu languages.[2]

Figure 2 displays the previous example pair but with the words split into syllables instead of morphemes. Note that long proper nouns cause oversegmentation in the syllabification in comparison to its corresponding morphological segmentation. Since we have found this approach to work relatively well, we have, for the time being, decided not to segment the English texts morphologically.

[1] Turkish is also agglutinative.

[2] Some disjunctively written languages such as Northern Sotho (Griesel et al., 2010) and Tswana (Wilken et al., 2012), where the written words resemble syllables, have been involved in machine translation projects.

Figure 2: An example of an English-Zulu alignment before and after syllabification.

2. Data and preparation

For our experiments, we have attempted to obtain at least two different types of parallel text. Free-for-use Zulu-English texts are not so easy to find online, but eventually, we have chosen a marked-up version of the New Testament of the Bible (English: King James)[3] as well as the South African constitution of 1996.[4].

The Bible corpus is aligned on verse level. The fact that there are no abbreviations simplified the task of sentence splitting, which we deemed necessary since the length of verses may be too long for proper processing, especially in the case where words are split into morphemes. Therefore, we wrote and implemented a naive sentence splitter which assumes the lack of abbreviations.

Basic cleanup of the corpora was performed. In the Zulu Bible, we removed some extra text, as well as all double quotation marks, since they were not present in the English version. For English, we changed the first few verses to line up precisely with its Zulu counterpart to facilitate sentence alignment. In the constitution, we corrected some encoding issues. We also deleted some false translations and dealt with formatting-related issues such as tables which we removed.

Next, we used the above sentence splitter since the constitution also hardly contains any abbreviations. We then tokenized all the texts using the script *tokenizer.perl* which is distributed with Moses (Koehn et al., 2007), assuming no abbreviations.

Next, the sentence aligner Hunalign (Varga et al., 2005) was used to automatically align sentences. No dictionary was provided for the alignment process. In an attempt to ensure good quality output, only alignments with a probability score of 0.8 or above was used. We evaluated the alignment quality of a 5% sample (116 segments) of the aligned constitution and found only two problematic segments. In the first case, the sentence splitting was incorrect, whereas in the second case, a 10-token clause was omitted in the translation. We therefore feel confident that the alignments are of high quality.

While constructing a gold standard (see section 3.) we found that the alignment quality of the Bible corpus was poor. As such, we have decided not to use this for any quantitative evaluations. We suspect that differences in sentence composition, such as the handling of compound sentences (full stops or semi-colons in English versus commas in Zulu) have played a role.

Our last pre-processing step before invoking automatic word alignment was to segment the Zulu text into syllables. A very simple implementation was used where the end of the syllable is always assumed to be a vowel. This is a known rule in Zulu with few exceptions, such as in the case of loan words.[5]

Tables 1 and 2 show some statistics for each of the corpora.

3. Word alignment experiments and construction of gold standards

We invoked the unsupervised word aligner MGIZA++ (Gao and Vogel, 2008) on the sentence-aligned sentences. The output of both directions of alignment was combined with a selection of the heuristics as implemented in Moses (Koehn et al., 2007). Using this approach, we constructed a number of alignment sets, one for each method applied: *src2tgt* (source to target), *tgt2src* (target to source), *intersect* (intersection), *union*, *grow*, *grow-diag*, *grow-diag-final* and *grow-diag-final-and*. The set of *grow* heuristics are designed to balance precision and recall and work by iteratively adding links to the set of intersection alignments starting at neighbouring lexical units. For example, *grow-diag* focuses more on precision whereas *grow-diag-final* focuses more on recall. *src2tgt* refers to the asymmetrical source-to-target alignments of MGIZA++ where a source-side unit may only have one alignment but a target-side unit may have multiple, and with *tgt2src*, it is the other way around.[6]

Next, we proceeded to create small alignment gold standards for our corpora. Unfortunately, as mentioned before, the sentence alignment for the Bible corpus proved to be insufficient. Therefore, our gold standard only consisted of text from the constitution.

Our tool of choice was Handalign,[7] a tool for which, among other options, a graphical user interface can be used for the alignment of lexical units. We proceeded to correct output from the automatic alignments as combined with the *intersection* heuristic alignments, as this seemed like the method with the least amount of work. For this work, we did not make distinctions between high-confidence (*good*) and lower-confidence (*fuzzy*) alignments, although this would certainly be possible in the future.

As manual word alignment is non-trivial, we set about following a set of guidelines which we attempted to implement as consistently as possible. A few issues remain which we might address again in the future, depending on their influence on extrinsic evaluation tasks. For this experiment, we have decided to align as many units as possible for the facilitation of statistical machine translation. However, we still

[3] http://homepages.inf.ed.ac.uk/s0787820/bible/

[4] http://www.polity.org.za/polity/govdocs/constitution/

[5] One example of such an exception in the text is the Zulu word for *Sanskrit*, *isiSanskrit*, which was sillabified as *i si Sa nskri t*.

[6] We refer the reader to the following URL for more information: http://www.statmt.org/moses/?n=FactoredTraining.AlignWords

[7] http://www.cs.utah.edu/hal/HandAlign/

	Bible	Constitution	All
Sentence count	13154	3091	16245
Post-alignment sentence count	2245	2321	4566
Post-alignment/pre-syllabification token count	20628	33828	54456
Post-alignment/post-syllabification token count	58773	100619	159392

Table 1: Data statistics for the Zulu corpora. Sentence and token counts are valid for the texts after initial cleaning.

	Bible	Constitution	All
Sentence count	12535	3143	15678
Post-alignment sentence count	2245	2321	4566
Post-alignment token count	37244	45000	82244

Table 2: Data statistics for the English corpora. Sentence and token counts are valid for the texts after initial cleaning.

keep untranslated units with extra information, which may result in bad or unnecessary translations, unaligned. This includes function words and syllables. Additionally:

- In the case of non-literal translations, but for which clear boundaries still exist, such as between single words and phrases, the lexical units are still aligned. For example: *seat* (*of Parliament*) → *indawo yokuhlala* (literally: *place of sitting*)
- Where explicit counterparts for syntactic arguments exist, they are aligned. When, in Zulu, they are repeated in the form of syllabic morphemes, but no similar anaphor exists in English, we keep them unaligned. For example, in the case of *Money may be withdrawn* → *Imali ingakhishwa*, the first prefix *I-* is aligned with *Money* along with *-mali*. However, the subject concord in *ingakhishwa* (*i-*) which refers back to *Imali*, is not aligned. We have arrived at this decision based on the fact that such concords should align with English pronouns if present, but not with the antecedent noun. We thought that it would seem inconsistent if we decided to align the concord with the English noun only if the pronoun is not present. In the light of this, we have made the decision to not align the Zulu concords with the English nouns at all.
- In the case of phrases for which the segmentation into syllables and words makes no semantic sense (i.e. is too fine-grained), we attempt a simple and arbitrary monotic alignment. For example, with *take into account* → *bhe ke le le* and *Cape Town* → *Ka pa*, the word *take* is aligned with *bhe*, although the word *bhekelele* derived from the verb *bheka*, and *Cape* is aligned with *Ka* and *Town* with *pa*, although clearly no such distinctions exist.
- Where an English noun phrase of the form adj+noun was translated into a possessive noun phrase in Zulu, the possessive particle was not aligned. For example: *Electoral Commission* → *IKhomishani yokhetho* (literally: commission of voting). Here the syllable in the position of the possessive particle *yo-* was not aligned, since the English was not worded as a possessive noun phrase.
- Where an English noun phrase was translated with a Zulu noun phrase containing a relative clause, the relative prefix and optional suffix (-yo) were only aligned if an obvious English counterpart existed, such as *that* in the following example: *following words* → *amagama alandelayo* (literally: *words that follow*). Here the prefix *a-* and the suffix *-yo* are left unaligned as the English did not contain a relative clause.

For this work, we produced a small gold standard consisting of 20 sentence pairs. As this is too small to provide really meaningful quantitative results, we focus on a qualitative evaluation as a stepping stone to future alignment approaches.

4. Evaluation

Although we did not perform a quantitative evaluation of the Bible corpus, it may be worth noting that manual inspection suggests that proper nouns are frequently aligned succesfully. Eventually, this may prove to be useful for tasks such as named entity recognition or the compilation of proper name lexica.

The *tgttosrc* (target-to-source) combination heuristic only models one-to-many alignments from an English word to (possibly) multiple syllables in Zulu. A particularly interesting example of a successful alignment (even though with slight differences to our guidelines) is presented in figure 3. In this case the syllables of the noun *isakhiwo* (English: institution) are correctly aligned to the English noun, but also the cross alignments to the subject reference *si-* in *yisiphi* as well as the object reference *-si-* in *asinekeze* is correctly aligned.

The *intersection* heuristic provided the highest precision and lowest recall as expected. An interesting outcome from these alignments is that these alignments often selected the syllable in a Zulu noun or verb from the stem of the word. It therefore seems that this conservative heuristic is able to very accurately identify some kind of semantic "kernel" of the word:

- other → *enye* (aligned with *nye*)
- written → *esibhaliwe* (aligned with *bha*)

Figure 3: Example of automatic alignments generated by the *tgt2src* heuristic. This demonstrates the successful alignment of the noun *institution* with both the corresponding Zulu noun, as well as its corresponding subject and object *-si-* syllables. Both of these correspond to the proper morphological segmentation.

- person → *umuntu* (aligned with *ntu* — the monosyllabic noun stem)

The *union* alignment had the highest recall as expected. It also contained several incorrect long distance alignments and cross alignments.

Finally, for the sake of interest, we also provide precision, recall and F-score for the automatic word alignments as measured against the gold standard (Table 3).

The scores in relation to each are more or less expected. For example, *intersect* has the highest precision, *union* has the highest recall, while *grow-diag* has a higher precision but lower recall than *grow-diag-final*. However, the substantially higher recall of *tgt2src* in comparison with *src2tgt* is somewhat surprising. Although the high precision of *tgt2src* can be partly explained by the fact that the asymmetry of its alignment approximates our alignment approach, where a single English word is very often aligned to multiple Zulu syllables, it remains interesting that it almost has the highest F-score, beaten only by *grow-diag*. However, a larger gold standard is required to make any definite conclusions.

5. Future work

With the Zulu words now segmented into very fine constituent parts, the lack of similar segmentation in the English becomes more apparent. Although English is not agglutinative, some level of morphological analysis might still be useful. For example, past tense markers and plural suffixes are expected to align with certain syllables (or morphemes in the case of a morphological analysis). English prefixes such as *multi-*, *co-*, *re-*, *non-* are likely to find meaningful alignments with Zulu morphemes below the word level. Words with hyphens were not separated by the tokenizer. *Auditor-General*, *self-determination* and *full-time* are examples from the constitution corpus where simple splitting on hyphens could have made finer-grained alignment possible.

On the Zulu side, we would of course like to use an accurate morphological analyzer for the proper segmentation into morphological units. A promising candidate is ZulMorph (Pretorius and Bosch, 2003), which currently only outputs a list of candidate analyses.

In the long run, we hope to be able to create a larger gold standard comprising a variety of domains. With more training data, we should be able to train a decent machine translation system, although this certainly brings along its own set of challenges.

Another exciting prospect, especially considering the context of less-resourced languages, is the projection of English metadata such as POS tags and morpho-syntactic structure on the Zulu text in order to train taggers and parsers. For part-of-speech tagging, De Pauw et al. (2011) and Garrette et al. (2013) are among authors who have produced interesting work. For the projection of syntactic structure, see, for example, Colhon (2012).

6. Conclusion

Syllabification can be used successfully as a mostly language-independent method for word segmentation. For the task of word alignment, this facilitates more fine-grained word and morpheme alignment while not requiring the existence of a fully-trained morphological analyzer. Our work suggests that this can be applied succesfully to the English-Zulu language pair, requiring very little time and resources. We believe that this may provide opportunities for the faster development of resources and technologies for less-resourced languages, which includes the field of machine translation.

7. Data availability

For our experiments, we made use of data that are in the public domain. In the same spirit, we are making our processed data available under the Creative Commons Attribution-ShareAlike 4.0 licence (CC BY-SA 4.0). Please contact the authors for any inquiries.

8. References

Mehmet Talha Çakmak, Süleyman Acar, and Gülsen Eryigit. 2012. Word alignment for English-Turkish language pair. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2177–2180, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Mihaela Colhon. 2012. Language engineering for syntactic knowledge transfer. *Comput. Sci. Inf. Syst.*, 9(3):1231–1247.

Guy De Pauw, Peter Waiganjo Wagacha, and Gilles-Maurice Schryver. 2011. Exploring the SAWA corpus: collection and deployment of a parallel corpus English-Swahili. *Language Resources and Evaluation*, 45(3):331–344.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

	Precision	Recall	F-score
intersect	**0.94**	0.28	0.43
grow	0.78	0.57	0.66
grow-diag	0.74	0.66	**0.699666**
grow-diag-final	0.62	0.75	0.68
grow-diag-final-and	0.72	0.68	0.697795
union	0.58	**0.76**	0.66
src2tgt	0.61	0.30	0.41
tgt2src	0.67	0.73	0.699216

Table 3: Precision, recall and F-score against the gold standard. Note how extremely close the top F-scores are to each other, and the interesting difference in recall between *src2tgt* and *tgt2src*.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of POS-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013)*, pages 583–592, Sofia, Bulgaria, August.

M. Griesel, C. McKellar, and D. Prinsloo. 2010. Syntactic reordering as pre-processing step in statistical machine translation from English to Sesotho sa Leboa and Afrikaans. In F. Nicolls, editor, *Proceedings of the 21st Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, pages 205–210.

L.M. Hyman. 2003. Segmental phonology. In D. Nurse and G. Phillipson, editors, *The Bantu Languages*, pages 42–58. Routledge, New York.

Kimmo Kettunen, Paul McNamee, and Feza Baskaya. 2010. Using syllables as indexing terms in full-text information retrieval. In *Proceedings of the 2010 Conference on Human Language Technologies – The Baltic Perspective: Proceedings of the Fourth International Conference Baltic HLT 2010*, pages 225–232, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laurette Pretorius and Sonja E. Bosch. 2003. Finite-state computational morphology: An analyzer prototype for Zulu. *Machine Translation*, 18(3):195–216.

Laurette Pretorius and Sonja Bosch. 2009. Exploiting cross-linguistic similarities in Zulu and Xhosa computational morphology. In *Proceedings of the First Workshop on Language Technologies for African Languages*, AfLaT '09, pages 96–103, Stroudsburg, PA, USA. Association for Computational Linguistics.

Patti Spinner. 2011. Review article: Second language acquisition of Bantu languages: A (mostly) untapped research opportunity. *Second Language Research*, 27(3):418–430.

D. Varga, L. Németh, P. Halácsy, A. Kornai, V. Trón, and V. Nagy. 2005. Parallel corpora for medium density languages. In *Proceedings of Recent Advances in Natural Language Processing (RANLP-2005)*, pages 590–596.

I. Wilken, M. Griesel, and C. McKellar. 2012. Developing and improving a statistical machine translation system for English to Setswana: a linguistically-motivated approach. In A. De Waal, editor, *Proceedings of the 23rd Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*.

Computational Estonian Grammar in Grammatical Framework

Inari Listenmaa*, **Kaarel Kaljurand**†

*Chalmers University of Technology
Gothenburg, Sweden
inari@chalmers.se

†University of Zurich
Zurich, Switzerland
kaljurand@gmail.com

Abstract

This paper introduces a new free and open-source linguistic resource for the Estonian language — a computational description of the Estonian syntax and morphology implemented in Grammatical Framework (GF). Its main area of use is in controlled natural language applications, e.g. multilingual user interfaces to databases, but thanks to the recent work in robust parsing with GF grammars, it can also be used in wide-coverage parsing and machine translation applications together with other languages implemented as part of GF's Resource Grammar Library (RGL). In addition to syntax rules that implement all the RGL functions, this new resource includes a full paradigm morphological synthesizer for nouns, adjectives and verbs that works with 90%–100% accuracy depending on the number of input forms, as well as a general purpose monolingual lexicon of 80,000 words which was built from existing Estonian language resources.

Keywords: Estonian, Grammatical Framework

1. Introduction

Estonian is a Finnic language spoken by ∼1 million people, mostly in Estonia. This paper describes a new computational linguistic resource for Estonian. This resource is implemented in Grammatical Framework (GF) (Ranta, 2011) and contains a morphological synthesizer; a wide variety of syntactic functions, fully implementing the language-neutral API of the GF Resource Grammar Library (RGL) (Ranta, 2009); and a 80k-word lexicon built using existing Estonian language resources.

The GF framework allows a programmer to concisely and human-readably describe a (controlled) language of a domain, and is e.g. suitable for building multilingual and multimodal dialog systems (Bringert et al., 2005), as well as wide-coverage parsers (Angelov, 2011). The GF RGL is an implementation of the main syntactic classes (e.g. noun phrase) and functions (e.g. construction of clause from a transitive verb and two noun phrases) for 30 languages. As most of the RGL has a language-independent API, programming multilingual applications does not require native knowledge of the involved languages. Recent work in robust and probabilistic parsing with GF grammars (Angelov and Ljunglöf, 2014) , as well as in hybrid machine translation (Angelov et al., 2014), has opened up a new possibility of using the RGL as a core component in general purpose machine translation applications (Ranta, 2012).

The new Estonian resource grammar (RG) implements the RGL functions for Estonian, connecting Estonian to the other 29 languages implemented in the RGL and making it possible to add Estonian to any application written in GF. As for the syntactic functions and the representation of word classes, the Estonian RG follows largely the existing Finnish RG implementation. The outcome allows us to summarize the main differences between Estonian and Finnish, at least in the context of GF-style language description. The constructors for morphological paradigms were implemented from scratch, closely following the existing descriptions of Estonian morphology. Existing resources allow us to evaluate this implementation. The existing resources could also be used with relatively little effort to construct a large general purpose lexicon. The current implementation is available under the LGPL license and further maintained at `https://github.com/GF-Estonian/GF-Estonian`.

This paper is structured as follows: Section 2. discusses previous approaches to a computational Estonian grammar; Section 3. provides a general overview of GF; Section 4. describes the implementation of morphology; Section 5. describes the implementation of syntax; Section 6. describes the creation and the sources of a large Estonian lexicon; Section 7. describes a preliminary evaluation, focusing on the correctness of the implementation of morphology and syntax; and finally Section 8. describes future work.

2. Related work

Estonian has a well-studied computational morphology implemented in three independent systems: the morphology software of the Institute of the Estonian Language[1], ESTMORF (Kaalep, 1997), finite-state morphology (Uibo, 2005; Pruulmann-Vengerfeldt, 2010). In addition to general morphological rules that can handle unknown words, these systems also include large lexicons of irregular words. The ESTMORF system has been commercialized as a set of morphology tools by the company Filosoft[2] and is available as spellers and thesauri in mainstream text processing suites. In computational syntax, the focus has been

[1] `http://www.eki.ee/tarkvara/`
[2] `http://www.filosoft.ee`

on wide-coverage shallow parsing, specifically morphological disambiguation and detection of the main syntactic tags (subject, object, premodifier, etc.), implemented in the Constraint Grammar framework, resulting in Estonian Constraint Grammar (EstCG) (Müürisep et al., 2003). This shallow syntactic analysis is being used as the basis in further experiments with full dependency parsing (Bick et al., 2005). Various lexical resources containing both morphosyntactic and semantic information have been developed, notably the Estonian WordNet (EstWN) (Vider and Orav, 2002) and various verb frame lexicons (Rätsep, 1978; Kaalep and Muischnek, 2008; Müürisep et al., 2003). Most of these existing computational resources can be almost directly used for the purposes of our grammar, either for the bootstrapping of lexicons or as gold standard in various evaluations.

The Estonian RG was greatly influenced by the existing GF RGL grammars, especially the Finnish RG (Ranta, 2008), as among the RGL languages, Finnish is by far the closest to Estonian. Our implementation started by copying the Finnish syntax rules and internal representations, which we then gradually changed to reflect Estonian. Estonian and Finnish have been compared to each other and to other European languages e.g. in (Metslang, 2010; Metslang, 2009), showing that there are morphosyntactic features where Estonian has distanced itself from Finnish towards German and Russian. This work has highlighted the parts of our initial Finnish RG port that required a change.

3. Grammatical Framework

GF is a framework for building multilingual grammar applications. Its main components are a functional programming language for writing grammars and a resource library that contains the linguistic details of many natural languages. A GF program consists of an *abstract syntax* (a set of functions and their categories) and a set of one or more *concrete syntaxes* which describe how the abstract functions and categories are linearized (turned into surface strings) in each respective concrete language. The resulting grammar describes a mapping between concrete language strings and their corresponding abstract trees (structures of function names). This mapping is bidirectional — strings can be *parsed* to trees, and trees *linearized* to strings. As an abstract syntax can have multiple corresponding concrete syntaxes, the respective languages can be automatically *translated* from one to the other by first parsing a string into a tree and then linearizing the obtained tree into a new string.

The GF programming language is a grammar formalism based on Parallel Multiple Context-Free Grammars (Ljunglöf, 2004), and is optimized to handle natural language features like morphological variation, agreement, and long-distance dependencies. It supports various forms of modularity and convenience constructs such as regular expressions and overloaded operators, which generally enable the programmer to write concise and readable code, which can be later verified by a domain expert or a linguist. The compiled form of the grammar (PGF) can be embedded into applications via bindings to major programming languages (Angelov et al., 2010).

The purpose of the RGL is to contain the linguistic knowledge of a large number of different natural languages, allowing an application programmer to focus on the domain semantics of the application rather than the linguistic details of its languages. Most of the library has a language-independent API, but language-specific functions can be included as part of the *Extra* modules. Recent work on robust and probabilistic parsing with GF (Angelov, 2011; Angelov and Ljunglöf, 2014) has made the parser computationally less sensitive to highly ambiguous grammars and also tolerate unknown syntactic constructs. As a result, the RGL can be used as a wide-coverage grammar suitable for parsing unconstrained text[3]. This work also makes it possible to translate unconstrained texts between the languages of the RGL, provided that their resources fully implement the language-neutral part of the API and that they contain a large lexicon whose entries are aligned with the multilingual *Dictionary* module of 65k lemmas, currently implemented by 8 of the 30 languages in the RGL.

4. Morphology

Estonian is an inflective language: a declinable word has typically 28 or 40 different inflectional forms and a verb typically 47 forms. Estonian inflection involves appending inflectional affixes to a stem, as well as alternations in the stem itself. New Estonian words can be formed freely and productively by derivation and compounding.

Our implementation of the Estonian morphology is a set of constructor functions ("smart paradigms" in GF terminology) that allow the user to specify a lexical entry by providing its word class, one or more base forms, and the particle and compounding parts. Based on this information, the constructor generates the complete morphological paradigm. The number of input base forms can vary; the more information in the input, the more accurate the generated lexical entry.

4.1. Nouns

Nouns (Figure 1 shows their representation in the Estonian RG) inflect in 14 cases and 2 numbers, resulting in 28 forms.

```
Noun : Type = {s : NForm => Str} ;

param
  NForm  = NCase Number Case ;
  Number = Sg | Pl ;
  Case = Nom | Gen | Part
  | Illat | Iness | Elat | Allat | Adess | Ablat
  | Transl | Ess | Termin | Abess | Comit ;
```

Figure 1: Representation of nouns

The noun constructor requires at most 6 forms (singular nominative, genitive, partitive, illative, and plural genitive and partitive) to correctly build all 28 forms. To simplify lexicon building, we implemented 4 additional constructors that require 1–4 input forms. The implementation of the 1-arg constructor follows (Kaalep, 2012) which describes the rules underlying the system of open morphology, i.e.

[3]Demo in `http://cloud.grammaticalframework.org/wc.html`

rules that Estonian speakers apply to unseen words. The constructor guesses the other base forms by looking at the ending of the singular nominative and trying to estimate the position of syllable stress based on the vowel/consonant pattern. Additional constructors cover nouns with a non-default stem vowel (which becomes visible in genitive), nouns that end with 'e' and are derived from a verb (type VII in (Kaalep, 2012)), and various less regular words.

Using the 1-arg operator allows one to build a full form lexicon from a simple word list (where nouns are typically represented by their dictionary form of singular nominative). This list should also contain information on compound borders which can be effectively added using unsupervised tools such as Morfessor (Virpioja et al., 2013) that have been found to work well on e.g. Finnish.

Pronouns, adjectives and numerals inflect similarly to nouns and use the same constructors.

4.2. Adjectives

Adjectives (Figure 2) inflect like nouns in case and number, but have three degrees: positive, comparative and superlative. Typically they agree in case and number with the modified noun. However, adjectives derived from past participles are invariable as premodifiers, and there is a small set of adjectives (e.g. *valmis*, *eri*, *-karva*) which stay unchanged in all positions. We have introduced a 3-value parameter `Infl` to indicate the agreement type.

```
Adjective : Type = {
  s : Degree => AForm => Str ;
  infl : Infl
} ;

param
  AForm = AN NForm | AAdv ;
  Degree = Posit | Compar | Superl ;
  Infl = Regular | Participle | Invariable ;
```

Figure 2: Representation of adjectives

4.3. Verbs

Verbs (Figure 3) inflect in voice, mood, tense, person and number. In addition, there are non-finite forms and participles that inflect like nouns. In total, the inflection table in the grammar has 40 forms, out of which 11 are non-finite. A non-inflecting component of multi-word verbs, such as *aru saama* 'to understand', is stored in the `p` field.

All verbs except *olema* 'be' and *tulema* 'go' are built from 8 verb forms, and there are 19 templates to build these from one or two verb forms. We followed the classification from (Erelt et al., 2006) when building the templates, and the choice of the 8 forms was guided by (Erelt et al., 2007). In addition to the 8-argument constructor, we have implemented 1–4 argument smart paradigms, which apply a suitable template to build the 8 forms.

```
Verb : Type = {
  s  : VForm => Str ;
  p  : Str  --particle verbs
} ;

param
  VForm =
     Presn Number Person | Impf Number Person
   | Condit Number Person | Quotative Voice
   | Imper Number | Imperp3 | Imperp1Pl | ImpNegPl
   | PassPresn Bool | PassImpf Bool
   | PresPart Voice| PastPart Voice | Inf InfForm ;

  Voice = Act | Pass ;
  InfForm =
    InfDa | InfDes
  | InfMa | InfMas | InfMast | InfMata | InfMaks ;
```

Figure 3: Representation of verbs

5. Syntax

5.1. Modification

Noun phrases Adjectives generally precede the noun that they modify; postmodification is possible but quite marginal, and it is not implemented in this grammar. Adjective phrases inherit the three different agreement types of adjectives; see parameter `Infl` in Figure 2.

- Regular adjectives: agree in case and number in all degrees and positions.
- Past participles used as adjectives: do not agree as premodifier in positive degree (*väsinud mehele* tired:SG.NOM man-SG.ALL 'to the tired man'), agree in comparative and superlative, act like regular adjectives as predicative (*mees muutus väsinuks* 'man became tired-SG.TRANSL').
- Invariable adjectives: do not agree as premodifier (*valmis linnades* 'ready:SG.NOM town-PL.INE'), do not allow comparative and superlative, do not inflect as predicative (*linn sai valmis* 'town became ready:SG.NOM').

On a continuum between AP and NP modifiers, Estonian has two frequent and productive constructions. One is an open class of genitive attributes, such as placenames, that do not agree with the noun and cannot be used as predicatives:

(1) saksa autodes
German:SG.GEN car-PL.INE
'in German cars'

The other construction is where a noun inflected in a case modifies another noun:

(2) lillevaas on puust laual
flower-vase:NOM is wood-ELA table-ADE
'the flower vase is on a wooden table'

Both constructions are currently implemented as invariable adjective phrases, but could benefit from having a dedicated category in the *Extra* module. This would prevent overgeneration, e.g. using genitive attributes as predicatives, which is ungrammatical (**auto on saksa* 'the car is German:GEN').

In addition, we have implemented RGL functions for modifying noun phrases with relative clauses, apposition, possessive noun phrases, adverbs and question clauses; all of them as simple concatenation of fixed units before or after the noun.

Other types can be modified with adverbs and adverbial phrases. RGL has four types of adverbs: for adjectives (*very small*), numerals (*more than 5*), verbs (*always sings*) and verb phrases (*see a cat on the hill*). The position of an adverb in a clause depends on whether it attaches to a verb, verb phrase or the complete clause.

5.2. Complementation

Nouns and adjectives The RGL has categories `A2`, `A3`, `N2` and `N3` for nouns and adjectives that expect one or more arguments, e.g. *distance* [from something] [to something]. These categories are lifted to adjective phrases and noun phrases before they start interacting on the clause level, and their complement is fixed.

Verbs Verb requires a certain case from its subject and object(s). The morphological properties of the verb are shown in Figure 3, and the complement properties in Figure 4. The type of intransitive verb `V` adds a subject case (`sc` field); two- and three-place verbs `V2`, `VV`, etc. extend the intransitive verb by adding one or more complement fields (`c2`, `c3`, `vi`).

```
V = Verb ** {sc : NPForm} ; --subject case

V2, VA, V2Q, V2S =
      V ** {c2 : Compl} ;      --see it; become red
V2A = V ** {c2, c3 : Compl} ; --paint it black
VV =  V ** {vi : InfForm} ;    --start singing
V2V = V ** {c2 : Compl ;
            vi : InfForm} ;    --tell him to do
V3 =  V ** {c2, c3 : Compl} ; --give it to her
```

Figure 4: Verbs with complements

The most common object cases are genitive and partitive. Constructions with genitive as an object case have some special properties: the case changes in negation, imperative and certain non-finite complement clauses. However, for independent pronouns, the object case is partitive in these constructions, and it does not change. To handle this phenomenon, `NP` has a boolean `isPron` field to distinguish between the two origins, and the parameter `NPForm` has a special value `NPAcc`, which points to genitive for noun-based NPs and partitive for pronoun-based NPs.

Object case indicates also aspect; for many verbs, both genitive and partitive object are grammatical, expressing the perfective and imperfective state of the action. The GF solution is to add two verbs to the lexicon, one with genitive object and other with partitive object.

Estonian makes frequent use of multi-word verbs: roughly 20% of all the predicates used in the texts are multi-word (Kaalep and Muischnek, 2008). The implementation of verbs in the RG suits well constructions with an uninflecting component, although an inflecting component of a fixed phrase is only possible to be analyzed as a complement. There is ongoing work on extending the RGL with intermediate layers between syntax and semantics (Grūzītis et al., 2012), including better support for various types of multi-word expressions. These additions improve the quality of RGL-based translations, as well as making the writing of application grammars easier.

5.3. Comparison to Finnish

Out of 48 categories, 23 differ in the Finnish grammar, resulting mostly from morphological factors: the lack of vowel harmony, possessive suffixes and question clitics in Estonian made many categories simpler. Conversely, the categories for Estonian adjectives and adjective phrases are more complex due to different agreement types. Unlike in Finnish, verb phrase retains the field for the non-inflecting component of multi-word verbs, because its placement depends on the word order of the whole clause.

Major syntactic differences were in word order and modifier agreement. The Estonian word order is more variable, preferring verb-second order and allowing discontinuity between finite and non-finite verb in auxiliary constructions. Some of the most complex syntactic phenomena, such as the choice of object case, could be reused without modification from the Finnish grammar.

6. Lexicon

In the context of the resource grammar, a lexicon is a set of lexical functions (0-place functions) and their linearizations that make use of the morphology API, i.e. overloaded operators like `mkN` and `mkV2`. In order to construct the lexicon from an existing resource, it must contain information about the word class (such as noun, transitive verb), sufficiently many word forms to allow the morphological operators to construct the full paradigm, and information about the inherent features such as case requirements for complements of verbs. For multilingual applications (e.g. machine translation) the function identifiers should be shared with the lexicons for other RGL languages, i.e. they should contain language-independent identifiers.

We have automatically constructed a 80k-word lexicon from all the nouns, adjectives and adverbs in EstWN (v67); the verbs of the EstCG lexicon, which provides information on the verb complement and adjunct cases; and the database of multi-word verbs (Kaalep and Muischnek, 2008). Morfessor 2.0 was used for compound word splitting of nouns (as the input resources did not mark compound borders), and Filosoft's morphology tools were used to generate the base forms of nouns and adjectives (6 forms), and verbs (8 forms). The current lexicon does not distinguish word senses and does not map entries via language independent identifiers to the other large lexicons of RGL, but we expect that this information can be easily added by relying on the interlingual indices (ILI) of EstWN. Table 1 lists the types of lexical entries.

7. Evaluation

In order to generate a gold standard evaluation set for the morphological constructors, we processed the words in EstWN with Filosoft's morphology tools. For each word we preserved only the last component of a possibly compound word and then generated all its forms. Our constructors were then applied to the base forms and their output compared against the gold standard. An entry was considered to be correctly generated if all the corresponding forms matched.

#	Constructor pattern	Comment
33409	mkN (mkN "...")	common compound noun
27599	mkN "..."	common noun
10197	mkV "particle" (mkV "...")	multi-word (intransitive) verb
3402	mkV "..."	intransitive verb
3396	mkAdv "..."	adverb
3006	mkA (mkN "...")	adjective
492	mkV2 (mkV "...")	transitive verb with genitive (default) object
320	mkV2 (mkV "...") cpartitive	transitive verb with partitive object
18	mkVV (mkV "...")	verb with a verbal complement
81839		total

Table 1: Frequency distribution of constructor patterns in DictEst. "..." marks one or more input forms

Testset	Constr.	1-arg	2-arg	3-arg	4-arg
nouns	`mkN`	91.1	95.4	97.1	98.2
adjective	`mkN`	90.0	93.6	95.2	96.9
verbs	`mkV`	90.5	96.6	98.3	99.7

Table 2: The percentage of correct results for 1–4-argument constructors tested on 21734 nouns, 3382 (positive) adjectives, and 5959 verbs from EstWN.

The results (Table 2) show that 90% of the lexical entries can be constructed from only the lemma form, and that almost all the verb forms can be reliably generated from just 4 base forms. A similar evaluation for Finnish (Ranta, 2008) found the Finnish 1-arg noun constructor to be 80% accurate, indicating that Finnish morphology is slightly more complex. This is also reflected by the number of noun forms that the worst-case constructor needs: 6 for Estonian and 10 for Finnish. 40% of the words that are incorrectly processed by the 1-arg constructor are contained in the list of 10k most frequent Estonian lemmas (Kaalep and Muischnek, 2004), meaning that these words are probably irregular (and thus cannot be captured with general rules) but likely to be found in any general purpose lexicon.

In order to evaluate the Estonian implementation of the shared RGL API, we have linearized and verified all the example sentences (425) in the RGL online documentation[4]. Additionally, we have ported some of the existing GF applications (MOLTO Phrasebook (Ranta et al., 2012), ACE-in-GF (Camilleri et al., 2012)) to Estonian. Although not covering all the constructs offered via the API, these applications give a good indication that the implementation of the syntactic functions is correct and represents the default Estonian utterances for the respective constructs.

8. Future work

As future work we want to better formalize the differences between Estonian and Finnish. The Estonian RG is currently an independent module in the RGL, but the GF language and the RGL design offer a clean way for sharing code and expressing differences via *parametrized modules* (also known as *functors*) and *Diff*-modules. Using a functor makes the code less repetitive (existing Romance and Scandinavian functors have achieved up to 80% sharing) and can offer an empirical measure of language similarity (Prasad and Virk, 2012). A *Finnic* functor would also simplify a possible addition of other Finnic languages, such as Võro and Karelian—most of them very scarce in language technology resources. Additional ways to formally compare GF-implemented languages is to look at the complexity and the predictive power of the morphological operators (Détrez and Ranta, 2012), and the parsing speed and complexity (Angelov, 2011).

Another future direction is to use the Estonian RG in wide coverage parsing and machine translation applications, similarly to the recent work on robust and probabilistic parsing with GF (Angelov, 2011; Angelov and Ljunglöf, 2014). The main additional requirement for such a system is to map the currently monolingual lexicon to the multilingual one. This process will be aided by WordNet ILI codes, but needs some manual work in checking, and adding words that are not from WordNet. For parsing applications, an Estonian-specific probability assignment to RGL tree structures is needed.

This paper evaluated the correctness of smart paradigms and syntactic functions. A more thorough evaluation that looks at the syntactic and lexical coverage, parsing speed, readability of the code etc. (see also the goals listed in (Ranta, 2009)) is also left as future work.

9. References

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

Angelov, Krasimir, Bringert, Björn, and Ranta, Aarne. (2010). PGF: A Portable Run-time Format for Type-theoretical Grammars. *Journal of Logic, Language and Information*, 19(2):201–228. 10.1007/s10849-009-9112-y.

Angelov, Krasimir, Bringert, Björn, and Ranta, Aarne. (2014). Speech-Enabled Hybrid Multilingual Translation for Mobile Devices. *European Chapter of the Association for Computational Linguistics, Gothenburg*.

[4] `http://www.grammaticalframework.org/lib/doc/synopsis.html`

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

Bick, Eckhard, Uibo, Heli, and Müürisep, Kaili. (2005). Arborest — a Growing Treebank of Estonian. *Nordisk Sprogteknologi*.

Bringert, Björn, Ljunglöf, Peter, Ranta, Aarne, and Cooper, Robin. (2005). Multimodal Dialogue System Grammars. *Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue*.

Camilleri, John J., Fuchs, Norbert E., and Kaljurand, Kaarel. (2012). Deliverable D11.1. ACE Grammar Library. Technical report, MOLTO project, June. `http://www.molto-project.eu/biblio/deliverable/ace-grammar-library`.

Détrez, Grégoire and Ranta, Aarne. (2012). Smart paradigms and the predictability and complexity of inflectional morphology. In *EACL (European Association for Computational Linguistics)*, Avignon, April. Association for Computational Linguistics.

Erelt, Tiiu, Leemets, Tiina, Mäearu, Sirje, and Raadik, Maire. (2006). *Eesti õigekeelsussõnaraamat*.

Erelt, Tiiu, Erelt, Mati, and Ross, Kristiina. (2007). *Eesti keele käsiraamat*.

Grūzītis, Normunds, Paikens, Peteris, and Barzdins, Guntis. (2012). Framenet resource grammar library for gf. In *CNL*, pages 121–137.

Kaalep, Heiki-Jaan and Muischnek, Kadri. (2004). Frequency Dictionary of Written Estonian of the 1990ies. In *The First Baltic Conference. Human Language Technologies — the Baltic Perspective*, pages 57–60.

Kaalep, Heiki-Jaan and Muischnek, Kadri. (2008). Multiword verbs of Estonian: a database and a corpus. pages 23–26.

Kaalep, Heiki-Jaan. (1997). An Estonian morphological analyser and the impact of a corpus on its development. *Computers and the Humanities*, 31(2):115–133.

Kaalep, Heiki-Jaan. (2012). Eesti käänamissüsteemi seaduspärasused. *Keel ja Kirjandus*, 6:418–449.

Ljunglöf, Peter. (2004). *Expressivity and Complexity of the Grammatical Framework*. Ph.D. thesis, University of Gothenburg.

Metslang, Helle. (2009). Estonian grammar between Finnic and SAE: some comparisons. *STUF-Language Typology and Universals*, 62(1-2):49–71.

Metslang, Helle. (2010). A General Comparison of Estonian and Finnish. Technical report.

Müürisep, Kaili, Puolakainen, Tiina, Muischnek, Kadri, Koit, Mare, Roosmaa, Tiit, and Uibo, Heli. (2003). A new language for constraint grammar: Estonian. In *International Conference Recent Advances in Natural Language Processing*, pages 304–310.

Prasad, K.V.S. and Virk, Shafqat. (2012). Computational evidence that hindi and urdu share a grammar but not the lexicon. In *3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP), collocated with COLING 12*.

Pruulmann-Vengerfeldt, Jaak. (2010). Praktiline lõplikel automaatidel põhinev eesti keele morfoloogiakirjeldus. Master's thesis.

Ranta, Aarne, Enache, Ramona, and Détrez, Grégoire. (2012). Controlled Language for Everyday Use: the MOLTO Phrasebook. In *Proceedings of the Second Workshop on Controlled Natural Language (CNL 2010)*, Lecture Notes in Computer Science. Springer.

Ranta, Aarne. (2008). How predictable is Finnish morphology? An experiment on lexicon construction. In Nivre, J., Dahllöf, M., and Megyesi, B., editors, *Resourceful Language Technology: Festschrift in Honor of Anna Sågvall Hein*, pages 130–148. University of Uppsala.

Ranta, Aarne. (2009). The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2(2).

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

Ranta, Aarne. (2012). Machine translation and type theory. In Dybjer, P., Lindström, Sten, Palmgren, Erik, and Sundholm, G., editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 281–311. Springer Netherlands.

Rätsep, Huno. (1978). *Eesti keele lihtlausete tüübid*. Valgus.

Uibo, Heli. (2005). Finite-state morphology of Estonian: Two-levelness extended. In *Proceedings of RANLP*, pages 580–584.

Vider, Kadri and Orav, Heili. (2002). Estonian wordnet and lexicography. In *Symposium on Lexicography XI. Proceedings of the Eleventh International Symposium on Lexicography*, pages 549–555.

Virpioja, Sami, Smit, Peter, Grönroos, Stig-Arne, and Kurimo, Mikko. (2013). Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland.

FST Trimming: Ending Dictionary Redundancy in Apertium

Matthew Marting, Kevin Brubeck Unhammer

St. David's School, Kaldera språkteknologi AS
Raleigh, NC., Stavanger, Norway
∅, unhammer+apertium@mm.st

Abstract

The Free and Open Source rule-based machine translation platform Apertium uses Finite State Transducers (FST's) for analysis, where the output of the analyser is input to a second, *bilingual* FST. The bilingual FST is used to translate analysed tokens (lemmas and tags) from one language to another. We discuss certain problems that arise if the analyser contains entries that do not pass through the bilingual FST. In particular, in trying to avoid "half-translated" tokens, and avoid issues with the interaction between multiwords and tokenisation, language pair developers have created redundant copies of monolingual dictionaries, manually customised to fit their language pair. This redundancy gets in the way of sharing of data and bug fixes to dictionaries between language pairs. It also makes it more complicated to reuse dictionaries outside Apertium (e.g. in spell checkers). We introduce a new tool to *trim* the bad entries from the analyser (using the bilingual FST), creating a new analyser. The tool is made part of Apertium's lttoolbox package.

Keywords: FST, RBMT, dictionary-redundancy

1. Introduction and background

Apertium (Forcada et al., 2011)[1] is a rule-based machine translation platform, where the data and tools are released under a Free and Open Source license (primarily GNU GPL). Apertium translators use Finite State Transducers (FST's) for morphological analysis, bilingual dictionary lookup and generation of surface forms; most language pairs[2] created with Apertium use the lttoolbox FST library for compiling XML dictionaries into binary FST's and for processing text with such FST's. This paper discusses the problem of redundancy in monolingual dictionaries in Apertium, and introduces a new tool to help solve it.

The following sections give some background on how FST's fit into Apertium, as well as the specific capabilities of lttoolbox FST's; then we delve into the problem of monolingual and bilingual dictionary mismatches that lead to redundant dictionary data, and present our solution.

1.1. FST's in the Apertium pipeline

Translation with Apertium works as a pipeline, where each *module* processes some text and feeds its output as input to the next module. First, a surface form like 'fishes' passes through the **analyser** FST module, giving a set of analyses like `fish<n><pl>/fish<vblex><pres>`, or, if it is unknown, simply `*fishes`. Tokenisation is done during analysis, letting the FST decide in a left-right longest match fashion which words are tokens. The compiled analyser technically contains several FST's, each marked for whether they have entries which are tokenised in the regular way (like regular words), or entries that may separate other tokens, like punctuation. Anything that has an analysis is a token, and any other sequence consisting of letters of the `alphabet` of the analyser is an unknown word token. Anything else can separate tokens.

After analysis, one or more **disambiguation** modules select which of the analyses is the correct one. The **pretransfer** module does some minor formal changes to do with multiwords.

Then a disambiguated analysis like `fish<n><pl>` passes through the **bilingual** FST. Using English to Norwegian as an example, we would get `fisk<n><m><pl>` if the bilingual FST had a matching entry, or simply `@fish<n><pl>` if it was unknown in that dictionary. So a known entry may get changes to both lemma (`fish` to `fisk`) and tags (`<n><pl>` to `<n><m><pl>`) by the bilingual FST. When processing input to the bilingual FST, it is enough that the *prefix* of the tag sequence matches, so a bilingual dictionary writer can specify that `fish<n>` goes to `fisk<n><m>` and not bother with specifying all inflectional tags like number, definiteness, tense, and so on. The tag suffix (here `<pl>`) will simply be carried over.

The output of the bilingual FST is then passed to the **structural transfer** module (which may change word order, ensure determiner agreement, etc.), and finally a **generator** FST which turns analyses like `fisk<n><m><pl>` into forms like 'fiskar'. Generation is the reverse of analysis; the dictionary which was compiled into a generator for Norwegian can also be used as an analyser for Norwegian, by switching the compilation direction.

A major feature of the lttoolbox FST package is the support for multiwords and compounds, and the automatic tokenisation of all *lexical units*. A lexical unit may be

- a simple, non-multi-word like the noun 'fish',
- a space-separated word like the noun 'hairy frogfish', which will be analysed as one token even though it contains a space, but will otherwise have no formal differences from other words,
- a multiword with *inner inflection* like 'takes out'; this is analysed as `take<vblex><pri><p3><sg># out` and then, after disambiguation, but before

[1] `http://wiki.apertium.org/`

[2] A *language pair* is a set of resources to translate between a certain set of languages in Apertium, e.g. Basque–Spanish.

bilingual dictionary lookup, turned into `take# out<vblex><pri><p3><sg>` – that is, the uninflected part (called the lemq) is moved onto the lemma,

- a token which is actually two words like 'they'll'; this is analysed as `prpers<prn><subj><p3><mf><pl> +will<vaux><inf>` and then split after disambiguation, but before bilingual dictionary lookup, into `prpers<prn><subj><p3><mf><pl>` and `will<vaux><inf>`,

- a combination of these three multiword types, like Catalan 'creure-ho que', analysed as `creure<vblex><inf> +ho<prn><enc><p3><nt># que` and then moved and split into `creure# que<vblex><inf>` and `ho<prn><enc><p3><nt>` after disambiguation, but before bilingual dictionary lookup.

In addition to the above multiwords, where the whole string is explicitly defined as a path in the analyser FST, we have dynamically analysed compounds which are not defined as single paths in the FST, but still get an analysis during lookup. To mark a word as being able to form a compound with words to the right, we give it the 'hidden' tag `<compound-only-L>`, while a word that is able to be a right-side of a compound (or a word on its own) gets the tag `<compound-R>`. These hidden tags are not shown in the analysis output, but used by the FST processor during analysis. If the noun form 'frog' is tagged `<compound-only-L>` and 'fishes' is tagged `<compound-R>`, the lttoolbox FST processor will analyse 'frogfishes' as a single compound token `frog<n><sg>+fish<n><pl>` (unless the string was already in the dictionary as an explicit token) by trying all possible ways to split the word. After disambiguation, but before bilingual dictionary lookup, this compound analysis is split into two tokens, so the full word does not need to be specified in either dictionary. This feature is very useful for e.g. Norwegian, which has very productive compounding.

1.2. The Problem: Redundant data

Ideally, when a monolingual dictionary for, say, English is created, that dictionary would be available for reuse unaltered (or with only bug fixes and additions) in all language pairs where one of the languages is English. Common data files would be factored out of language pairs, avoiding redundancy, giving *data decomposition*. Unfortunately, that has not been the case in Apertium until recently.

If a word is in the analyser, but not in the bilingual translation dictionary, certain difficulties arise. As the example above showed, if 'fishes' were unknown to both dictionaries, the output would be `*fishes`, while if it were unknown to only the second, the output of the analyser would be `@fish<n><pl>`, and of the complete translation just `@fish`. Given '*fishes', a post-editor who knows both languages can immediately see what the original was, while the half-translated `@fish` hides the inflection information in the source text. Just lemmatising the source text – which removes features like number, definiteness or tense – can skew meaning. For example, if the input were the Norwegian Bokmål 'Sonny gikk til hundene', "Sonny went to the dogs" meaning "Sonny's life went to ruin", a Norwegian Nynorsk translator seeing 'Sonny gjekk til @hund' would have to look at the larger context (or the original) to infer that it was the idiomatic meaning, and not "Sonny went to (a) dog". Similarly, in the Bokmål sentence 'To milliarder ødela livet', literally "Two billions [money] destroyed the life", the definite form is used to mean "his life"; since the lemma of 'livet' is ambiguous with the plural, the half-translated 'To milliarder ødela @liv' could just as well mean "Two billions destroyed lives".[3] But it gets worse: Some languages inflect verbs for *negation*, where the half-translated lemma would hide the fact that the meaning is negative. When translating from Turkish, if you see '@öl', you would definitely need more context or the original to know whether it was e.g. 'öldürdü' "s/he/it killed" or 'öldürmedi' "s/he/it did not kill" – if the MT is used for gisting, where the user doesn't know the source language, this becomes particularly troublesome. For single-token cases like this, a workaround for the post-editor is to carry surface form information throughout the pipeline, but as we will see below, this fails with multiwords and compounds, which are heavily used in many Apertium language pairs.

A word not known to the bilingual FST might not have its tags translated (or translated correctly) either. When the transfer module tries to use the half-translated tags to determine agreement, the *context* of the half-translated word may have its meaning skewed as well. For example, some nouns in Norwegian have only plural forms; when translating a singular form from into Norwegian, the bilingual FST typically would the singular tag to a plural tag for such words. Structural transfer rules insert determiners which are inflected for number; if the noun were half-translated (thus with the singular tag of the input), we would output a determiner with the wrong number, so the errors in that word may give errors in context as well.

Trying to write transfer rules to deal with half-translated tags also *increases the complexity of transfer rules*. For example, if any noun can be missing its gender, that's one more exception to all rules that apply gender agreement, as well as any feature that interacts with gender. This matters even more where tagsets differs greatly, and bilingual dictionaries are used to translate between tagsets. For example, the Northern Sámi analyser, being primarily developed outside Apertium, has almost no tags in common with the Norwegian analyser; so the bilingual dictionary transfer e.g. the sequence `<V><TV><Ind><Prs><Sg1>` into `<vblex><pers><pres><sg><p1>`, for any word that actually has a match in the bilingual dictionary. If it doesn't have a match, tags of course cannot be transferred. Structural transfer rules expect to deal with the Norwegian tagset. If a word enters structural transfer with only the Sámi tags, we need to write exceptions to all transfer rules to deal with the possibility that tags may be in another

[3] Examples from `http://www.bt.no/bergenpuls/litteratur/Variabelt-fra-Nesbo-3081811.html` and `http://www.nettavisen.no/2518369.html`.

tagset.

Then there are the issues with tokenisation and multiwords. Multiwords are entries in the dictionaries that may consist of what would otherwise be several tokens. As an example, say you have 'take' and 'out' listed in your English dictionary, and they translate fine in isolation. When translating to Catalan, we want the phrasal verb 'take out' to turn into a single word 'treure', so we list it as a multiword with *inner inflection* in the English dictionary. This makes any occurrence of forms of 'take out' get a single-token multiword analysis, e.g. 'takes out' gets the analysis `take<vblex><pri><p3><sg># out`. But then the whole multiword *has* to be in the bilingual dictionary if the two words together are to be translated. If another language pair using the same English dictionary has both 'take' and 'out' in its bilingual dictionary, but not the multiword, the individual words in isolation will be translated, but whenever the whole string 'take out' is seen, it will only be lemmatised, not translated. This is both frustrating for the language pair developer, and wasted effort in the case where we don't *need* the multiword translation.

Assuming we could live with some frustration and transfer complexity, we could try to carry input surface forms throughout the pipeline to at least avoid showing lemmatised forms. But here we run into another problem. Compounds and multiwords that consist of several lemmas are split into two units before transfer, e.g. the French contraction 'au' with the analysis `à<pr>+le<det><def><m><sg>` turns into `à<pr>` and `le<det><def><m><sg>`, while the Norwegian compound 'vasskokaren' analysed as `vatn<n><nt><sg><ind><cmp>+kokar<n><m><sg><def>` turns into `vatn<n><nt><sg><ind><cmp>` and `kokar<n><m><sg><def>` – compounds may also be split at more than one point. We split the analysis so that we avoid having to list every possible compound in the bilingual dictionary, but we can't split the form. We don't even know which part of the form corresponds to which part of the analysis (and of course we would not want to translate half a word). If we attach the full form to the first part and leave the second empty, we run into trouble if only the second part is untranslatable, and vice versa. One hack would be to attach the form to the first analysis and let the second analysis instead have some special symbol signifying that it is the analysis of the form of the previous word, e.g. `vasskokaren/vatn<n><nt><sg><ind><cmp>` and `$1/kokar<n><m><sg><def>`. But now we need the bilingual FST to have a memory of previously seen analyses, which is a step away from being finite-state. It would also mean that other modules which run between the analyser and the bilingual FST in the pipeline have to be careful to avoid changing such sequences,[4] introducing brittleness into an otherwise robust system.

Due to such issues, most language pairs in Apertium have a separate copy of each monolingual dictionary, manually *trimmed* to match the entries of the bilingual dictionary; so in the example above, if 'take out' did not make sense to have in the bilingual dictionary, it would be removed from the copy of the monolingual dictionary. This of course leads to a lot of redundancy and duplicated effort; as an example, there are currently (as of SVN revision 50180) twelve Spanish monolingual dictionaries in stable (SVN trunk) language pairs, with sizes varying from 36,798 lines to 204,447 lines.

The redundancy is not limited to Spanish; in SVN trunk we also find 10 English, 7 Catalan, and 4 French dictionaries. If we include unreleased pairs, these numbers turn to 19, 28, 8 and 16, respectively. In the worst case, if you add some words to an English dictionary, there are still 27 dictionaries which miss out on your work. The numbers get even worse if we look at potential new language pairs. Given 3 languages, you only need $3 * (3 - 1) = 6$ monolingual dictionaries for all possible pairs (remember that a dictionary provides both an analyser and a generator). But for 4 languages, you need $4 * (4 - 1) = 12$ dictionaries; if we were to create all possible translation pairs of the 34 languages appearing in currently released language pairs, we would need $34 * (34 - 1) = 1122$ monolingual dictionaries, where 34 ought to be enough.

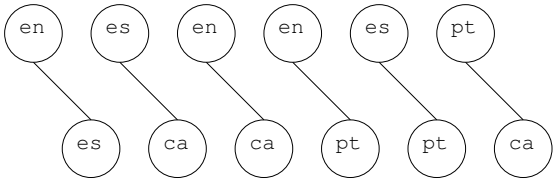

Figure 1: Current number of monodixes with pairs of four languages

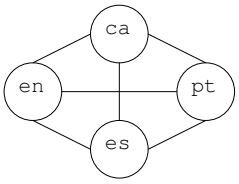

Figure 2: Ideal number of monodixes with four languages

The lack of shared monolingual dictionaries also means that other monolingual resources, like disambiguator data, is not shared, since the effort of copying files is less than the effort of letting one module depend on another for so little gain. And it complicates the reuse of Apertium's extensive (Tyers et al., 2010) set of language resources for other systems: If you want to create a speller for some language supported by Apertium (now possible for lttoolbox dictionaries via HFST (Pirinen and Tyers, 2012)), you either have to manually merge dictionaries in order to gain from all the work, or (more likely) pick the largest one and hope it's good enough.

1.3. A Solution: Intersection

However, there is a way around these troubles. Finite state machines can be intersected with one another to produce a

[4] E.g. the rule based disambiguation system Constraint Grammar allows for moving/removing analyses; and an experimental module for handling discontiguous multiwords also moves words.

new finite state machine. In the case of the Apertium transducers, what we want is to intersect the output (or **right**) side of the full analyser with the input (or **left**) side of the bilingual FST, producing a *trimmed* FST. We call this process *trimming*.

Some recent language pairs in Apertium use the alternative, Free and Open Source FST framework HFST (Linden et al., 2011).[5][6] Using HFST, one can create a "prefixed" version of the bilingual FST, this is is the concatenation of the bilingual FST and the regular expression `.*`, i.e. match any symbol zero or more times.[7] Then the command `hfst-compose-intersect` on the analyser and the prefixed FST creates the FST where only those paths of the analyser remain where the right side of the analyser match the left side of the bilingual FST. The prefixing is necessary since, as mentioned above, the bilingual dictionary is underspecified for tag suffixes (typically inflectional tags such as definiteness or tense, as opposed to lemma-identifying tags such as part of speech and noun gender).

The HFST solution works, but is missing many of the Apertium-specific features such as different types of tokenisation FST's, and it does not handle the fact that multiwords may split or change format before bilingual dictionary lookup. Also, unlike lttoolbox, most of the Apertium dictionaries compiled with HFST represent compounds with an optional transition from the end of the noun to the beginning of the noun dictionary – so if `frog<n>` and `fish<n>` were in the analyser, but `fish<n>` were missing from the bilingual FST, `frog<n>+fish<n>` would remain in the trimmed FST since the prefix `frog<n>.*` matches. In addition, using HFST in language pairs whose data are all in lttoolbox format would introduce a new (and rather complex) dependency both for developers, packagers and users who compile from source.

Thus we decided to create a new tool within lttoolbox, called `lt-trim`. This tool should trim an analyser using a bilingual FST, creating a trimmed analyser, and handle all the lttoolbox multiwords and compounds, as well as letting us retain the special tokenisation features of lttoolbox. The end result should be the same as perfect manual trimming. The next section details the implementation of `lt-trim`.[8]

2. Implementation of `lt-trim`

The implementation consists of two main parts: preprocessing the bilingual dictionary, and intersecting it with the analyser.

2.1. Preprocessing the bilingual dictionary

Like monolingual dictionaries, bilingual ones can actually define several FST's, but in this case the input is already tokenised – the distinction is only useful for organising the source, and has no effect on processing. So the first preprocessing step is to take the union of these FST's. This is as simple as creating a new FST F, with epsilon (empty/unlabelled) transitions from F's initial state, to each initial state in the union, and from each of their final states to F's final state.

Next, we append loopback transitions to the final state. Like mentioned in section 1.1. above, the bilingual dictionary is underspecified for tags. We want an analyser entry ending in `<n><pl>` to match the bilingual entry ending in `<n>`. Appending loopback transitions to the final state, ie. `<n>.*`, means the intersection will end up containing `<n><pl>`; we call the bilingual dictionary *prefixed* when it has the loopback transitions appended. The next section explains the implementation of prefixing.

The final preprocessing step is to give multiwords with inner inflection the same format as in the analyser. As mentioned in section 1.1., the analyser puts tags after the part of the lemma corresponding to the inflected part, with the uninflected part of the multiword lemma coming last.[9] The bilingual dictionary has the uninflected part before the tags, since it has to allow tag prefixes instead of requiring full tag sequences. Section 2.3. details how we move the uninflected part after the (prefixed) tags in preprocessing the bilingual dictionary.

2.2. Prefixing the bilingual dictionary

The lttoolbox FST *alphabets* consist of symbol pairs, each with a left (serves as the input in a transducer) and right (output) symbol. Both the pairs and the symbols themselves, which can be either letters or tags, are identified by an integer. First, we loop through analyser symbol-pairs, collecting identifiers of those where the right-side is a tag. Then for all these tags, we add new symbol-pairs with that tag on both sides to the bilingual FST.

These are used to create loopbacks in the bilingual FST using the function $\mathrm{appendDotStar}$. Transitions are created from the final states of the bilingual transducer that loop directly back with each of the identifiers. If we call the set of tag symbol-pairs captured from the analyser T, and the bilingual FST B, the prefixed bilingual FST is $BT*$ (in the next section, we write `.*` to avoid confusion with letter symbols).

2.3. Moving uninflected lemma parts

To turn `take# out<vblex>.*` into `take<vblex>.*# out` and so on, we do a depth-first traversal looking for a transition labelled with the # symbol. Then we replace the #-transition t with one into the new transducer returned by $\mathrm{copyWithTagsFirst(t)}$, which in this case would return `<vblex>.*# out`.

This function traverses the FST from the target of t (in the example above, the state before the space), building up two new transducers, new and lemq. Until we see the first tag, we add all transitions found to lemq, and record in a search state which was the last lemq state we saw. In the example above, we would build up a lemq transducer containing

[5] `http://hfst.sourceforge.net/`

[6] Partly due to available data in that formalism, partly due to features missing from lttoolbox like *flag diacritics*.

[7] In this article, we use the regular POSIX format for regular expression; the Xerox format used in HFST-compiled dictionaries differs.

[8] Available in the SVN version of the lttoolbox package, see `http://wiki.apertium.org/wiki/Installation`, the code itself is at `https://svn.code.sf.net/p/apertium/svn/trunk/lttoolbox`.

[9] Having the tags "lined up with" or at least close to the inflection they represent eases dictionary writing.

out (with an initial space). Upon seeing the first tag, we start adding transitions from the initial state of new.
When reaching a final tag state s, we add it and the last seen lemq state l to a list of pairs f. After the traversal is done, we loop through each (s, l) in f, creating a temporary copy of lemq where the lemq-state l has been made the only final state, and adding a #-transition from each final tag state s in new into that copy. In the example, we would make a copy of lemq where the state after the letter t were made final, and insert that copy after the final state s, the state after the <vblex>.*.[10]
Finally, we return new from copyWithTagsFirst and look for the next # in B.

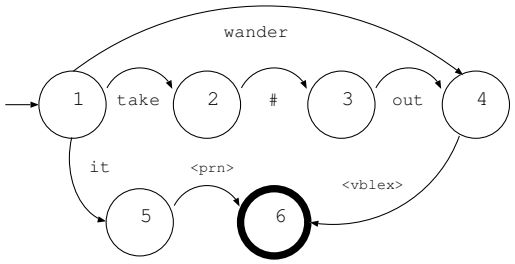

Figure 3: Input bilingual FST (letter transitions compressed to single arcs)

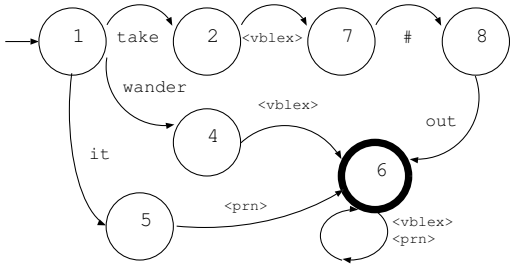

Figure 4: Fully preprocessed bilingual FST; analyses take<vblex># out and even take<vblex>+it<prn># out would be included after trimming on this

2.4. Intersection

The first method we tried of intersecting FST's consisted of multiplying them. This is a method that is simple to prove correct; however, it was extremely inefficient, requiring a massive amount of memory. First, the states of each transducer were multiplied, giving the cartesian product of the states. Every possible state pair, consisting of a state from the monolingual dictionary and the bilingual dictionary, was assigned a state in the trimmed transducer. Next, each of the transitions were multiplied. As the intersection is only concerned with the output of the monolingual dictionary and the input of the bilingual dictionary, the respective symbols had to match; very many of them did not, and a significant number of matching symbols resulted in transitions to redundant and unreachable states. These would be removed with minimisation, but the memory usage in the meantime made that method unusable for anything but toy dictionaries.
The tool now implements the much more efficient depth-first traversal method of intersection. Both the monolingual dictionary and the bilingual dictionary are traversed at the same time, in lockstep; only transitions present in both are further followed and added to the trimmed transducer.
The process is managed through noting which states are to be processed, the pair of states currently being processed, the next pair of states, and the states which have already been seen. Besides having reachable counterparts in both transducers, a state pair will only be added to the queue if it has not been seen yet.
However, to handle multiwords, a few other things are necessary.
If a + is encountered in the monolingual dictionary (indicating a +-type multiword) the traversal of the bilingual dictionary resumes from its beginning. In addition, in the event that a # is later encountered, the current position is recorded. The #-type multiwords alone can be easily handled if the bilingual dictionary is preprocessed to be in the same format as the monolingual dictionary, with the tags moved before the # as described above. However, a combination of both + and # requires that the traversal of the bilingual dictionary return to the state at which the + was first encountered.
We also need some exceptions for epsilon transitions; the idea here is that if we see an epsilon in one transducer, we move across that epsilon without moving in the other transducer, and vice versa. Compound symbols in the analyser are treated similarly: we add the transitions to the trimmed analyser and move forward in the analyser without moving forward in the bilingual FST.

2.5. `lt-trim` in use

The depth-first traversal method uses memory and time similar to regular compilation in lttoolbox. When testing on Norwegian Nynorsk–Norwegian Bokmål, with 64152 entries in the bilingual dictionary and 179369 in the full Bokmål monolingual dictionary, the trimming memory usage is lower than with monolingual compilation (both under 10 % on a 4 GB machine), and time with trimming is only 10 seconds compared to 20 seconds with monolingual compilation. With English–Catalan, with 27461 entries in the bilingual dictionary and 31784 in the Catalan dictionary (which also has more multiwords), the trimming memory usage is about double of the monolingual compilation (both still under 10 %) and time is slower with trimming, up from 9 to 13 seconds. We feel this is quite acceptable, and haven't made an effort to optimise yet.
To the end user (i.e. language pair developer), the tool is a simple command line program with three arguments: the input analyser FST, the input bilingual FST and the output

[10] If, from the original #, there were a lemq path that didn't have the same last-lemq-state (e.g. take# out<n>.* or even take# out.*) it would end up in a state that were not final after s, and the path would not give any analyses (such paths are removed by FST *minimisation*). But if a lemq path did have the same last state, we would want it included, e.g. take# part<vblex>.* to take<vblex>.*# part. Thus several lemq paths may lead from the # to the various first tag states, but we only connect those paths which were connected in the original bilingual dictionary.

trimmed analyser FST.

3. Ending Dictionary Redundancy

As mentioned in section 1.3., there are already language pairs in Apertium that have moved to a decomposed data model, using the HFST trimming method. At first, the HFST language pairs would also copy dictionaries, even if they were automatically trimmed, just to make them available for the language pair. But over the last year, we have created scripts for our GNU Autotools-based build system that let a language pair have a formal dependency on one or more monolingual data packages[11]. There is now an SVN module `languages`[12] where such monolingual data packages reside, and all of the new HFST-based languages pairs now use such dependencies, which are trimmed automatically, instead of making redundant dictionary copies. Disambiguation data is also fetched from the dependency instead of being redundantly copied.

But most of the released and stable Apertium language pairs use lttoolbox and still have dictionary redundancy. With the new `lt-trim` tool, it is finally possible to end the redundancy[13] for the pairs which use lttoolbox, with its tokenisation, multiword and compounding features, and without having to make those pairs dependent on a whole other FST framework simply for compilation.

The tool has only recently been released, and there is still much work to do in converting existing language pairs to a decomposed data model. Monolingual dictionaries have to be merged, and the various language pairs may have altered the tag sets in more or less subtle ways that can affect disambiguation, transfer and other parts of the pipeline.

To give an example, merging the Norwegian Bokmål dictionaries of the language pairs Northern Sámi-Bokmål and Nynorsk-Bokmål (including changes to transfer and to the other involved dictionaries) took about three hours of work. However, this kind of merge work happened often in the past anyway when major changes happened to either dictionary; from now on it can be a one-time job. Future additions and changes to the common apertium-nob module will benefit both language pairs.

Any new languages added to Apertium can immediately reap the benefits of the tool, without this manual merge work; this goes for many of the in-development pairs too (e.g. the English-Norwegian language pair now depends on the Bokmål and Nynorsk monolingual package).

4. Conclusion

In this article, we have presented a new tool to trim monolingual dictionaries to fit the data in Apertium language pairs. The tool has already been implemented and used in several language pairs, starting the process to end monolingual dictionary redundancy.

In future, we plan to research how to deal with compounds when trimming HFST dictionaries, as well as further merging monolingual dictionaries.

Acknowledgements

Part of the development was funded by the Google Code-In[14] programme. Many thanks to Francis Tyers and Tommi Pirinen for invaluable help with the development of `lt-trim`.

5. References

Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Krister Linden, Miikka Silfverberg, Erik Axelson, Sam Hardwick, and Tommi Pirinen, 2011. *HFST—Framework for Compiling and Applying Morphologies*, volume Vol. 100 of *Communications in Computer and Information Science*, pages 67–85.

Tommi A Pirinen and Francis M Tyers. 2012. Compiling apertium morphological dictionaries with hfst and using them in hfst applications. In G. De Pauw, G-M de Schryver, M.L. Forcada, K. Sarasola, F.M. Tyers, and P.W. Wagacha, editors, *[SALTMIL 2012] Workshop on Language Technology for Normalisation of Less-Resourced Languages*, pages 25–28, May.

Antonio Toral, Mireia Ginestí-Rosell, and Francis Tyers. 2011. An Italian to Catalan RBMT system reusing data from existing language pairs. In F. Sánchez-Martínez and J.A. Pérez-Ortiz, editors, *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 77–81, Barcelona, Spain, January.

Francis M. Tyers, Felipe Sánchez-Martínez, Sergio Ortiz-Rojas, and Mikel L. Forcada. 2010. Free/Open-Source Resources in the Apertium Platform for Machine Translation Research and Development. *Prague Bull. Math. Linguistics*, 93:67–76.

[11] This both means that source and compiled monolingual files are made available to the make files of the language pair, and that the configure script warns if monolingual data packages are missing. Packagers should be able to use this so that if a user asks their package manager, e.g. `apt-get`, to install the language pair `apertium-foo-bar`, it would automatically install dependencies `apertium-foo` and `apertium-bar` first, and use files from those packages.

[12] `http://wiki.apertium.org/wiki/Languages`

[13] One could argue that there is still *cross-lingual* redundancy in the bilingual dictionaries – Apertium by design does not use an interlingua. Instead, the Apertium dictionary crossing tool `crossdics` (Toral et al., 2011) provides ways to extract new translations during development: Given bilingual dictionaries between languages A-B and B-C, it creates a new bilingual dictionary between languages A-C. One argument for not using an interlingua during the translation process is that the dictionary resulting from automatic crossing needs a lot of manual cleaning to root out false friends, unidiomatic translations and other errors – thus an interlingua would have to contain a lot more information than our current bilingual dictionaries in order to automatically disambiguate such issues. It would also require more linguistics knowledge of developers and heighten the entry barrier for new contributors.

[14] `https://code.google.com/gci/`

Shallow-transfer rule-based machine translation for the Western group of South Slavic languages

Hrvoje Peradin, Filip Petkovsky, Francis M. Tyers

University of Zagreb, University of Zagreb, Institut for språkvitskap
Faculty of Science, Faculty of Electrical Engineering, Det humanistiske fakultet
Dept. of Mathematics, and Computer Science, N-9037 Universitetet i Tromsø
hperadin@gmail.com, filip.petkovski@fer.hr, ftyers@prompsit.com

Abstract

The South Slavic languages, spoken mostly in the Balkans, make up one of the three Slavic branches. The South Slavic branch is in turn comprised of two subgroups, the Eastern subgroup containing Macedonian and Bulgarian, and the western subgroup containing Serbo-Croatian and Slovenian. This paper describes the development of a bidirectional machine translation system for the western branch of South-Slavic languages — Serbo-Croatian and Slovenian. Both languages have a free word order, are highly inflected, and share a great degree of mutual inteligibility. They are also under-resourced as regards free/open-source resources. We give details on the resources and development methods used, as well as an evaluation, and general directions for future work.

1. Introduction

The South Slavic language branch, which is spoken mostly in the Balkans, makes up one of the three Slavic branches. The South Slavic branch itself is in turn comprised of two subgroups, the Eastern subgroup containing Macedonian and Bulgarian, and the western subgroup containing Serbo-Croatian and Slovenian.

The Serbo-Croatian (hbs)[1] dialects are the native language of most people in Serbia, Croatia, Montenegro and Bosnia and Herzegovina. They were formed on the basis of the *štokavian* dialects which got their name from the form *što* (or *šta*), which is used for the interrogative pronoun 'what?'. A second group of dialects from the Serbo-Croatian language group is the Čakavian group spoken in western Croatia, Istria, the coast of Dalmatia, and some islands in the Adriatic. Like the štokavian dialects, the *čakavian* dialects got their name from the form *ča* used for the same interrogative pronoun. Finally, the third main group of Serbo-Croatian dialects, spoken in north-western Croatia, uses *kaj* instead of *što*, and is called *kajkavian*. An intermediate dialect between Serbo-Croatian, Bulgarian and Macedonian is the Torlakian dialect. The three or four standardised varieties of Serbo-Croatian are all based on the štokavian dialect.

Slovenian (slv) is the native language of Slovenia, and is also spoken in the neighbouring areas in Italy and Austria. While Slovenian has many different dialects, it shares some features with the Kajkavian and Čakavian dialects spoken in Croatia. Although the speakers of the different Serbo-Croatian dialects can understand each other without any serious difficulties, a Serbo-Croatian speaker can have a difficult time understanding a speaker of a Slovenian dialect.

[1]We use the term 'Serbo-Croatian' as an abbreviation for Bosnian-Croatian-Montenegrin-Serbian.

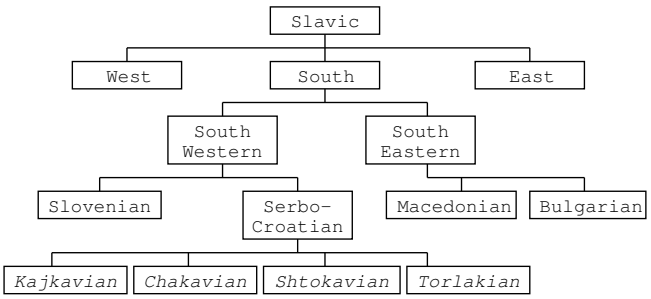

Figure 1: A traditional division of the South-Slavic languages. All four standard varieties of Serbo-Croatian (Bosnian, Croatian, Montenegrin, and Serbian) are based on the štokavian dialect.

2. Design

2.1. The Apertium platform

The Apertium[2] platform (Forcada et al., 2011) is a modular machine translation system. The typical core layout consists of a letter transducer morphological lexicon.[3] The transducer produces cohorts[4] which are then subjected to a morphological disambiguation process. Disambiguated readings are then looked up in the bilingual dictionary, which gives the possible translations for each reading. These are then passed through a lexical-selection module (Tyers et al., 2012), which applies rules that select the most appropriate translation for a given source-language context. After lexical selection, the readings, which are now pairs of source and target language lexical forms are passed through a syntactic transfer module that performs word reordering, deletions, insertions, and basic syntactic chunking. The final module is another letter transducer which generates sur-

[2]http://wiki.apertium.org/

[3]A list of ordered pairs of word surface forms and their lemmatised analyses.

[4]A cohort consists of a surface form and one or more readings containing the lemma of the word and the morphological analysis.

	Bosnian	Croatian	Montenegrin	Serbian
Čakavian	-	-i-,-e-,-je-	-	-
Kajkavian	-	-e-,-ie-,-ei,-i-	-	-
Štokavian	-ije-,-je-	-ije-,-je-,-i-	-ije-,-je-	-e-,-ije-,-je-
Torlakian	-	-	-	-e-

Table 1: Intersection of Serbo-Croatian languages and dialects. All four standard variants are based on the štokavian dialect, but other dialects are considered to *belong* to a standard. The entries in the table correspond to the *yat* reflex.

face forms in the target language from the bilingual transfer output cohorts.

2.2. Constraint Grammar

This language pair uses a Constraint Grammar (CG) module[5] for disambiguation (Karlsson, 1995). The CG formalism consists of hand-written rules that are applied to a stream of tokens. Depending on the morphosyntactic context of a given token the rules select or exclude readings of a given surface form, or assign additional tags.

3. Development

3.1. Resources

This language pair was developed with the aid of on-line resources containing word definitions and flective paradigms, such as *Hrvatski jezični portal*[6] for the Serbo-Croatian side. For the Slovenian side we used a similar online resource *Slovar slovenskega knjižnega jezika*,[7] and the *Amebis Besana* flective lexicon.[8]

The bilingual dictionary for the language pair was developed from scratch, using the *EUDict*[9] online dictionary and other online resources.

3.2. Morphological analysis and generation

The basis for this language pair are the morphological lexicons for Serbo-Croatian (from the language pair Serbo-Croatian–Macedonian, `apertium-hbs-mak`) and Slovenian (from the language pair Slovenian–Spanish, `apertium-slv-spa`) (Table 2). Both lexicons are written in the XML format of *lttoolbox*[10] (Ortiz-Rojas et al., 2005), and were developed as parts of their respective language pairs, during the Google Summer of Code 2011.[11] Since the lexicons had been developed using different frequency lists, and slightly different tagsets, they have been further trimmed and updated to synchronise their coverage.

3.3. Disambiguation

Though for both languages there exists a number of tools for morphological tagging and disambiguation (Vitas and Krstev, 2004; Agić et al., 2008; Šnajder et al., 2008; Peradin and Šnajder, 2012), there were none freely available when the work described in this paper was carried out (Summer, 2012). Likewise, the adequately tagged corpora at the time were mostly non-free (Erjavec, 2004; Tadić, 2002).

Due to the high degree of morphological variation the two languages exhibit, given the lack of a tagged corpus, and the relatively small lexicon size of our analysers, we were unable to obtain satisfactory results with the statistical tagger canonically used in Apertium language pairs. For this reason we chose to use solely Constraint Grammar (CG) for disambiguation. The CG module does not provide complete disambiguation, so in the case of any remaining ambiguity the system picks the first output analysis. Due to the similarities between the languages, we were able to reuse some of the rules developed earlier for Serbo-Croatian.

3.4. Lexical transfer

The lexical transfer was done with an *lttoolbox* letter transducer composed of bilingual dictionary entries.

In addition to standard word-by-word pairing of translations, additional paradigms were added to the transducer to handle less general tagset mismatches, that were deemed more convenient to be resolved directly. However, to minimize manual correction of tag mismatches, most of the tagset differences were handled with macros in the structural transfer module.

The bilingual dictionary was also used to introduce tags for tricky cases such as when the adjective comparison is synthetic on one side, and analytic on the other (e.g. *zdravije* vs. *bolj zdravo*). This difference is arbitrary, and unfortunately every such occurence needed to be marked by hand. These tags were expanded to the correct form later in structural transfer.

3.5. Lexical selection

Since there was no adequate and free Slovenian – Serbo-Croatian parallel corpus, we chose to do the lexical selection relying only on hand-written rules in Apertium's lexical selection module (Tyers et al., 2012). For cases not covered by our hand-written rules, the system would choose the default translation from the bilingual dictionary. We provide examples of such lexical selection rules.

Phonetics-based lexical selection: many words from the Croatian and Serbian dialects differ in a single phoneme. An example are the words *točno* in Croatian and *tačno* in Serbian (engl. *accurate*). Such differences were solved through the lexical selection module using rules like:

```
<rule>
  <match lemma="točno" tags="adv.*">
```

[5] Implemented in the CG3 formalism, using the `vislcg3` compiler, available under GNU GPL. For a detailed reference see: `http://beta.visl.sdu.dk/cg3.html`

[6] `http://hjp.srce.hr`

[7] `http://bos.zrc-sazu.si/sskj.html`

[8] `http://besana.amebis.si/pregibanje/`

[9] `http://eudict.com/`

[10] `http://wiki.apertium.org/wiki/Lttoolbox`

[11] `http://code.google.com/soc/`

```
    <select lemma="točno" tags="adv.*"/>
  </match>
</rule>
```

for Croatian, and

```
<rule>
  <match lemma="točno" tags="adv.*">
    <select lemma="tačno" tags="adv.*"/>
  </match>
</rule>
```

for Serbian and Bosnian.
Similarly, the Croatian language has the form *burza* (meaning stock exchange in English), while Serbian and Bosnian have *berza*. For those forms the following rules were written:

```
<rule>
  <match lemma="borza" tags="n.*">
    <select lemma="burza" tags="n.*"/>
  </match>
</rule>
```

for Croatian, and

```
<rule>
  <match lemma="borza" tags="n.*">
    <select lemma="berza" tags="n.*"/>
  </match>
</rule>
```

for Serbian and Bosnian.
Another example of a phonetical difference are words which have h in Croatian and Bosnian, but v in Serbian. Such words include *kuha* and *duhan* in Croatian and Bosnian, but *kuva* and *duvan* in Serbian. Similar rules were written for the forms for *porcelain* (procelan in Serbian and porculan in Croatian), *salt* (so and sol) and so on.
While the Serbian dialect accepts the Ekavian and Ikavian reflexes, the Croatian dialect uses only the Ijekavian reflex. Since the selection for the different reflexes of the *yat* vowel is done in the generation process, no rules were needed in the lexical selection module.
Internationalisms have been introduced to Croatian and Bosnian mainly through the Italian and German languages whereas they have entered Serbian through French and Russian. As a result, the three dialects have developed different phonetic patterns for international words.
Examples of rules for covering such varieties include:

```
<rule>
  <match lemma="Betlehem" tags="np.*">
    <select lemma="Betlehem" tags="np.*"/>
  </match>
</rule>
```

for Croatian and Bosnian, and

```
<rule>
  <match lemma="Betlehem" tags="np.*">
    <select lemma="Vitlejem" tags="np.*"/>
  </match>
</rule>
```

for Serbian.
Finally, the Croatian months used for the Gregorian calendar have Slavic-derived names and differ from the original Latin names. For example, the Croatian language has the word *siječanj* for *January*, and the Serbian language has the word *januar*. These differences were also covered by the lexical selection module. The number of disambiguation, transfer and lexical selection rules are shown in Table 3.

3.6. Syntactic transfer

Serbo-Croatian and Slovenian are very closely related, and their morphologies are very similar. Most of the transfer rules are thus macros written to bridge the notational differences in the tagsets, or to cover different word orders in the languages.
Following are examples of transfer rules we have written, that also illustrate some contrastive characteristics of the languages. The original rules are written in Apertium's XML DSL[12], and their syntax is quite lenghty. For the sake of brevity and clarity we give the rules in a more compact descriptive form:

- the future tense:

  (1) Gledal bom ↔ Gledat ću[13]
  [watch.LP.M.SG][be.CLT.P1.SG] ↔
  [watch.INF][will.CLT.P1.SG]
  (I will watch.)

  Both languages form the future tense in an analytic manner. While Slovenian uses a perfective form of the verb *to be* combined with the l-participle (analogous to Serbo-Croatian future II), Serbo-Croatian uses a cliticised form of the verb *to want* combined with the infinitive. Unlike the infinitive, the l-participle carries the information on the gender and number. Since in this simplest form we have no way of inferring the gender of the subject in the direction Serbo-Croatian → Slovenian the translation defaults to masculine.

- *lahko* and *moći*:

  (2) Bolezni lahko povzročijo virusi ↔ Bolesti mogu prouzročiti virusi
  [Diseases.ACC] [easily.ADV] [cause.P3.SG] [viruses.NOM] → [Diseases.ACC] [can.P3.SG] [cause.INF] [viruses.NOM]
  (Viruses can cause diseases.)

  Unlike its Serbo-Croatian cognate *lako* the adverb *lahko* in Slovenian, when combined with a verb has an additional meaning of *can be done*, expressed in Serbo-Croatian with the modal verb *moći*. Rules that cover these type of phrases normalise the target verb to infinitive, and transfer grammatical markers for number and person to the verb *moći*.

- *lahko* and conditional:

  (3) Lahko bi napravili ↔ Mogli bi napraviti
  [easily.ADV] [would.CLT.CND] [do.LP.PL] → [Can.LP.PL] [would.CLT.CND.P3.SG] [do.INF]
  (We/they could do)

[12] http://wiki.apertium.org/wiki/A_long_introduction_to_transfer_rules

[13] The Serbo-Croatian analyser covers both orthographical variants of the encliticised future tense (gledat ću / gledaću) as well.

Dictionary	Paradigms	Entries	Forms
Serbo-Croatian	1,033	13,206	233,878
Slovenian	1,909	13,383	147,580
Bilingual	69	16,434	–

Table 2: Statistics on number of lexicon entries for each of the dictionaries in the system.

Type	hbs→slv	slv→hbs
Disambiguation	194	28
Lexical selection	–	42
Transfer	47	98

Table 3: Statistics on the number of rules in each direction. For the lexical selection rules, the number indicates that there are 42 rules for each of the three standard varieties currently supported.

Another morphological difference is found in the conditional mood. The conditional marker in Serbo-Croatian is the aorist form of the verb *to be*, and carries the information on person and number[14]. Slovenian, and the majority of colloquial Serbo-Croatian varieties, use a frozen clitic form of the same verb.[15] Thus in cases like this example, when it is impossible to exactly infer the person and number the system defaults to the colloquial form.

- *lahko* and conditional more complicated:

  (4) Mi bi lahko napravili ↔ Mi bismo mogli napraviti

  [We.P1.PL] [would.CLT.CND] [easily.ADV] [do.LP.PL] → [We.P1.PL] [would.CLT.CND.P3.PL] [can.LP.PL] [do.INF]

  (We could do)

  The information on person and number is available on the pronoun *mi*, and can be copied in translation to the conditional verb.

- *treba* adverb to verb

  (5) je treba narediti → treba učiniti

  [is] [needed.ADV] [to be done.INF] → [needs.VB.P3.SG] [to be done.INF]

  (It needs to be done)

  Phrases with Slovenian adverb *treba* translate to Serbo-Croatian with the verb *trebati*. In its simplest form the phrase just translates as 3rd person singular.

  For the opposite direction *trebati* translates as the analogous verb *potrebovati*, so that no loss of morphological information occurs.

  (6) trebaju našu solidarnost → potrebujejo našu solidarnost

  (They need our solidarity)

  More complicated examples with different tenses and verb phrases involve word reodering:

  (7) narediti je bilo treba ↔ trebalo je napraviti

  [do.INF] [is.CLT.P3.SG] [was.LP.NT] [need.ADV] → [needed.LP.NT] [is.CLT.P3.SG] [do.INF]

  (It needed to be done.)

Language	SETimes	Europarl
Serbo-Croatian	85.41%	–
Slovenian	–	95.50%

Table 4: Naïve coverage

4. Evaluation

This sections covers the evaluation of the developed system. The system was tested by measuring the lexical coverage, and by performing a qualitative and a quantitative evaluation.

Lexical coverage was tested using existing free corpora, while the quantitative evaluation was performed on 100 postedited sentences (with 1,055 words in total) from the Slovenian news portal Delo. [16]

Statistics on the size of the resulting lexicons are given in table 2, and the rule counts are listed in table 3. While the lexicons are evenly matched, the number of rules is slightly in favour of the hbs side. This is due to the fact that after the initial development phase additional work has been done in the transfer module for the slv → hbs direction, and the disambiguation and lexical selection modules have been developed by native speakers of Serbo-Croatian who are not fluent in Slovene.

4.1. Lexical coverage

Coverage for the Serbo-Croatian–Slovenian language pair was measured using both the SETimes (Tyers and Alperen, 2010) and Europarl (Koehn, 2005) corpora. We measured coverage naively, meaning that we assume a word is in our dictionaries if at least one of its surface forms is found in the corpus. We are aware of the shortcomings of such an evaluation framework, however we decided to use it because of its simplicity.

The Serbo-Croatian → Slovenian side was evaluated using the SETimes corpus. As SETimes does not cover Slovenian the Slovenian → Serbo-Croatian side was evaluated only on the EuroParl corpus. The results are shown in table 4.

4.2. Quantitative

The quantitative evaluation was performed by 5 articles from the Slovenian news portal Delo. The articles were translated from Slovenian using Apertium, and were later corrected by a human post-editor in order to get a correct translation. The Word Error Rate (WER) was calculated by counting the number of insertions,

[14] *bih*, *bismo*, *biste*, or *bi*

[15] *bi* regardless of person and number

[16] http://www.delo.si/

substitutions and deletions between the post-edited articles and the original translation. We used the freely available `apertium-eval-translator` for calculating the WER and for bootstrap resampling (Koehn, 2004). We also reported the percentage of out of vocabulary words (OOV), and the total number of words per article. The results are given in table 5.

We also calculated both metrics for the output of Google Translate[17] and the results are presented in the same tables. Note that to compare the systems we made two posteditions, one from the Apertium translation, and the other from the Google translation, so as not to bias the evaluation in either direction.

The post-editting evaluation shows comparable results for our system and Google Translate according to WER and PER metrics. The Slovenian → Serbo-Croatian translation seems to be better than the Serbo-Croatian → Slovenian one which is due to the fact that more effort was put into developing the former direction.

4.3. Qualitative

The biggest problems are currently caused by the incompleteness of our dictionaries. The issues caused by OOV words are twofold. The less important issue is the fact that the system is unable to provide a translation for the unknown words — although in many cases, such as with proper names, these may result in *free rides*, that is the word is the same in both languages. However, the more important issue is that OOV words cause problems with disambiguation and transfer, since they break long chains of words into smaller ones and drastically reduce context information.

Next, we have seen that the number of disambiguation rules for Slovenian is not sufficient for high-quality disambiguation. The constraint grammar for the Slovenian side was written based on the constraint grammar for the Serbo-Croatian side, and it needs further work.[18]

We have also noticed difficulties in the transfer because of the loose grammar of both sides. Variations created by the free word order, and long distance relationships between sentence constituents make it difficult to write translation rules that cover a wide variety of language constructs. Adding additional rules does not significantly improve the performance of the system and OOV words make long transfer rules irrelevant.

Finally, because of the short timeframe, and due to the fact no reliable parallel corpus exists for this language pair,[19] we were unable to do much work on lexical selection. Our lexical-selection module is the least developed part of our system. We have not done any work on the Slovenian side and the number of rules for the Serbo-Croatian side is small.

5. Future work

The greatest difficulties for our system are caused by the long phrases present and the loose and free word order in the South Slavic languages. Because of that, in future we plan to put more effort into dealing with those problems. We are aware of the fact that it is difficult to write transfer rules between the two sides, and we intend to address that issue by first improving the coverage of our dictionaries.

After expanding the dictionaries, we intend to put more time into developing the Slovenian constraint grammar, and improve transfer by taking into account wider context.

We intend to work on more Slavic language pairs, including Serbo-Croatian–Russian, and improve our existing ones (see Figure 2), including Serbo-Croatian–Macedonian (Peradin and Tyers, 2012) using the resources and knowledge obtained by developing this language pair.

Finally, we will keep the resources up to date in regard to current regional linguistic developments, in particular we will add the Montenegrin language once the standard is completely agreed on.

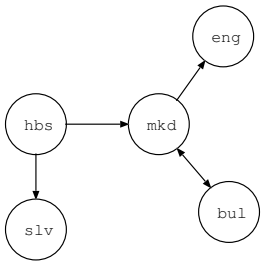

Figure 2: Language pairs including the South-Slavic languages in Apertium: `mkd` = Macedonian, `bul` = Bulgarian; `eng` = English.

6. Conclusions

This language pair was an encouraging take on a pair of closely related South-Slavic languages, and represents a satisfying conclusion to an MT chain of neighbouring languages (the pairs Serbo-Croatian–Macedonian and Macedonian–Bulgarian are also available in Apertium). While we are aware that it is still in its infancy, and has many flaws, it is a valuable free/open-source resource, and will serve as another solid ground for NLP in this language group.

Acknowledgements

The development of this language pair was funded as a part of the Google Summer of Code.[20] Many thanks to the language pair co-author Aleš Horvat and his mentor Jernej Vičič, and other Apertium contributors for their invaluable help and support.

7. References

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

T. Erjavec. 2004. MULTEXT-East version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth Int. Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1535–1538.

[17] `http://translate.google.com/`

[18] An evaluation of a more extensive Constraint grammar for Croatian can be found in (Peradin and Šnajder, 2012)

[19] There is e.g. `http://opus.lingfil.uu.se/`, but it consists mostly of texts from OpenSubtitles

[20] `http://code.google.com/soc/`

Article	Words	% OOV		WER	
		Apertium	Google	Apertium	Google
maraton	243	16.8	–	**[42.85, 47.92]**	[64.39, 74.56]
sonce	169	17.7	–	**[32.65, 45.33]**	[47.27, 58.62]
merkator	414	16.9	–	**[38.78, 48.14]**	[56.13, 70.30]
volitve	229	13.9	–	[37.81, 53.36]	[46.66, 62.67]
maraton	245	37.7	–	[52.78, 56.25]	[45.58, 63.87]
sonce	171	17.5	–	[47.50, 62.79]	[32.10, 58.49]
merkator	424	12.9	–	[45.78, 56.56]	[48.46, 64.15]
volitve	226	16.8	–	[47.00, 58.44]	[38.09, 58.10]

Table 5: Results for Word Error Rate (WER) in the Slovenian→Serbo-Croatian direction (top) and Serbo-Croatian→Slovenian (bottom). Scores in bold show a statistically significant improvement over the other system according to bootstrap resampling at $p = 0.95$.

M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

F. Karlsson. 1995. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4. Walter de Gruyter.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th MT Summit*, pages 79–86.

S.O. Ortiz-Rojas, M.L. Forcada, and G.R. Sánchez. 2005. Construcción y minimización eficiente de transductores de letras a partir de diccionarios con paradigmas. *Procesamiento de Lenguaje Natural*, 35:51–57.

H. Peradin and J. Šnajder. 2012. Towards a constraint grammar based morphological tagger for croatian. In *Text, Speech and Dialogue*, pages 174–182. Springer.

H. Peradin and F. M. Tyers. 2012. A rule-based machine translation system from Serbo-Croatian to Macedonian. In *Proceedings of the Third International Workshop on Free/Open-Source Rule-Based Machine Translation (FREERBMT12)*, pages 55–65.

Jan Šnajder, B Dalbelo Bašić, and Marko Tadić. 2008. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing & Management*, 44(5):1720–1731.

M. Tadić. 2002. Building the croatian national corpus. In *LREC2002 Proceedings, Las Palmas, ELRA, Pariz-Las Palmas*, volume 2, pages 441–446.

F. Tyers and M.S. Alperen. 2010. South-East European times: A parallel corpus of Balkan languages. In *Forthcoming in the proceedings of the LREC workshop on "Exploitation of multilingual resources and tools for Central and (South) Eastern European Languages*.

F. M. Tyers, F. Sánchez-Martínez, and M. L. Forcada. 2012. Flexible finite-state lexical selection for rule-based machine translation. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 213–220, Trento, Italy, May.

D. Vitas and C. Krstev. 2004. Intex and Slavonic morphology. *INTEX pour la linguistique et le traitement automatique des langues, Presses Universitaires de Franche-Comté*, pages 19–33.

Enhancing a Rule-Based MT System with Cross-Lingual WSD

Alex Rudnick[1], **Annette Rios**[2], **Michael Gasser**[1]
[1] School of Informatics and Computing, Indiana University
[2] Institute of Computational Linguistics, University of Zurich
{alexr,gasser}@indiana.edu, arios@ifi.uzh.ch

Abstract

Lexical ambiguity is a significant problem facing rule-based machine translation systems, as many words have several possible translations in a given target language, each of which can be considered a sense of the word from the source language. The difficulty of resolving these ambiguities is mitigated for statistical machine translation systems for language pairs with large bilingual corpora, as large n-gram language models and phrase tables containing common multi-word expressions can encourage coherent word choices. For most language pairs these resources are not available, so a primarily rule-based approach becomes attractive. In cases where some training data is available, though, we can investigate hybrid RBMT and machine learning approaches, leveraging small and potentially growing bilingual corpora. In this paper we describe the integration of statistical cross-lingual word-sense disambiguation software with SQUOIA, an existing rule-based MT system for the Spanish-Quechua language pair, and show how it allows us to learn from the available bitext to make better lexical choices, with very few code changes to the base system. We also describe Chipa, the new open source CL-WSD software used for these experiments.

Keywords: under-resourced languages, hybrid machine translation, word-sense disambiguation

1. Introduction

Here we report on the development of Chipa, a package for statistical lexical selection, and on integrating it into SQUOIA,[1] a primarily rule-based machine translation system for the Spanish-Quechua language pair. With very few code changes to SQUOIA, we were able to make use of the lexical suggestions provided by Chipa.

The integration enables SQUOIA to take advantage of any available bitext without significantly changing its design, and to improve its word choices as additional bitext becomes available. Our initial experiments also suggest that we are able to use unsupervised approaches on monolingual Spanish text to further improve results.

In this paper, we describe the designs of the Chipa and SQUOIA systems, discuss the data sets used, and give results on both how well Chipa is able to learn lexical selection classifiers in isolation, and to what extent it is able to improve the output of SQUOIA on a full Spanish-to-Quechua translation task.

In its current design, SQUOIA makes word choices based on its bilingual lexicon; the possible translations for a given word or multi-word expression are retrieved from a dictionary on demand. If there are several possible translations for a lexical item, these are passed along the pipeline so that later stages can make a decision, but if the ambiguity persists, then the first entry retrieved from the lexicon is selected. While there are some rules for lexical selection, they have been written by hand and only cover a small subset of the vocabulary in a limited number of contexts.

In this work, we supplement these rules with classifiers learned from Spanish-Quechua bitext. These classifiers make use of regularities that may not be obvious to human rule-writers, providing improved lexical selection for any word type that has adequate coverage in the training corpus.

[1] http://code.google.com/p/squoia/

Quechua is a group of closely related indigenous American languages spoken in South America. There are many dialects of Quechua; SQUOIA focuses on the Cuzco dialect, spoken around the Peruvian city of Cuzco. Cuzco Quechua has about 1.5 million speakers and some useful available linguistic resources, including a small treebank (Rios et al., 2009), also produced by the SQUOIA team.

2. SQUOIA

SQUOIA is a deep-transfer RBMT system based on the architecture of MATXIN (Alegria et al., 2005; Mayor et al., 2011). The core system relies on a classical transfer approach and is mostly rule-based, with a few components based on machine learning. SQUOIA uses a pipeline approach, both in an abstract architectural sense and in the sense that its pieces are instantiated as a series of scripts that communicate via UNIX pipes. Each module performs some transformation on its input and passes along the updated version to the next stage. Many modules focus on very particular parts of the representation, leaving most of their input unchanged.

In the first stages, Spanish source sentences are analyzed with off-the-shelf open-source NLP tools. To analyze the input Spanish text, SQUOIA uses FreeLing (Padró and Stanilovsky, 2012) for morphological analysis and named-entity recognition, Wapiti (Lavergne et al., 2010) for tagging, and DeSr (Attardi et al., 2007) for parsing. All of these modules rely on statistical models.

In the next step, the Spanish verbs must be disambiguated in order to assign them a Quechua verb form for generation: a rule-based module tries to assign a verb form to each verb chunk based on contextual information. If the rules fail to do so due to parsing or tagging errors, the verb is marked as ambiguous and passed on to an SVM classifier, which assigns a verb form even if the context of that verb does not unambiguously select a target form. This is among the most difficult parts of the translation process,

as the grammatical categories encoded in verbs differ substantially between Spanish and Quechua. In the next step, a lexical transfer module inserts all possible translations for every word from a bilingual dictionary. Then a set of rules disambiguates the forms with lexical or morphological ambiguities. However, this rule-based lexical disambiguation is very limited, as it is not feasible to cover all possible contexts for every ambiguous word with rules.
The rest of the system makes use of a classical transfer procedure. A following module moves syntactic information between the nodes and the chunks in the tree, and finally, the tree is reordered according to the basic word order in the target language. In the last step, the Quechua surface forms are morphologically generated through a finite state transducer.

3. CL-WSD with Chipa

Chipa is a system for cross-lingual word sense disambiguation (CL-WSD). [2] By CL-WSD, we mean the problem of assigning labels to polysemous words in source-language text, where each label is a word or phrase type in the target language.
This framing of word-sense disambiguation, in which we consider the possible senses of a source-language word to be its known target-language translations, neatly addresses the problem of choosing an appropriate sense inventory, which has historically been a difficult problem for the practical application of WSD systems (Agirre and Edmonds, 2006). Here the sense distinctions that the CL-WSD system should learn are exactly those that are lexicalized in the target language. The CL-WSD framing also sidesteps the "knowledge acquisition bottleneck" hampering other work in WSD (Lefever et al., 2011). While supervised CL-WSD methods typically require bitext for training, this is more readily available than the sense-annotated text that would otherwise be required.
To appreciate the word-sense disambiguation problem embedded in machine translation, consider for a moment the different senses of "have" in English. In *have a sandwich*, *have a bath*, *have an argument*, and even *have a good argument*, the meaning of the verb "to have" is quite different. It would be surprising if our target language, especially if it is not closely related, used a light verb that could appear in all of these contexts.
A concrete example for different lexicalization patterns in Spanish and Quechua are the transitive motion verbs: The Spanish lemmas contain information about the path of the movement, e.g. *traer* - 'bring (here)' vs. *llevar* - 'take (there)'. Quechua roots, on the other hand, use a suffix (-*mu*) to express direction, but instead lexicalize information about the manner of movement and the object that is being moved. Consider the following examples:

general motion verbs:

- *pusa-(mu-)*: 'take/bring a person'
- *apa-(mu-)*-: 'take/bring an animal or an inanimated object'

motion verbs with manner:

- *marq'a-(mu-)*: 'take/bring smth. in one's arms'
- *q'ipi-(mu-)*: 'take/bring smth. on one's back or in a bundle'
- *millqa-(mu-)*: 'take/bring smth. in one's skirts'
- *hapt'a-(mu-)*: 'take/bring smth. in one's fists'
- *lluk'i-(mu-)*: 'take/bring smth. below their arms'
- *rikra-(mu-)*: 'take/bring smth. on one's shoulders'
- *rampa-(mu-)*: 'take/bring a person holding their hand'

The correct translation of Spanish *traer* or *llevar* into Quechua thus depends on the context. Furthermore, different languages simply make different distinctions about the world. The Spanish *hermano* 'brother', *hijo* 'son' and *hija* 'daughter' all translate to different Quechua terms based on the person related to the referent; a daughter relative to her father is *ususi*, but when described relative to her mother, *warmi wawa* (Academia Mayor de La Lengua Quechua, 2005).
Chipa, then, must learn to make these distinctions automatically, learning from examples in available word-aligned bitext corpora. Given such a corpus, we can discover the different possible translations for each source-language word, and with supervised learning, how to discriminate between them. Since instances of a source-language word may be NULL-aligned, both in the training data and in actual translations, we allow users to request classifiers that consider NULL as a valid label for classification, or not, as appropriate for the application.
The software holds all of the available bitext in a database, retrieving the relevant training sentences and learning classifiers on demand. If a source word has been seen with multiple different translations, then a classifier will be trained for it. If it has been seen aligned to only one target-language type, then this is simply noted, and if the source word is not present in the training data, then that word is marked out-of-vocabulary. Memory permitting, these classifiers and annotations are kept cached for later usage. Chipa can be run as a server, providing an interface whereby client programs can request CL-WSD decisions over RPC.
Here classifiers are trained with the scikit-learn machine learning package (Pedregosa et al., 2011), using logistic regression (also known as "maximum entropy") with the default settings and the regularization constant set to $C = 0.1$. We also use various utility functions from NLTK (Bird et al., 2009).
For this work, we use familiar features for text classification: the surrounding lemmas for the current token (three on either side) and the bag-of-words features for the entire current sentence. We additionally include, optionally, the Brown cluster labels (see below for an explanation), both for the immediate surrounding context and the entire sentence. We suspect that more feature engineering, particularly making use of syntactic information and surface word forms, will be helpful in the future.

[2] Chipa the software is named for chipa the snack food, popular in many parts of South America. It is a cheesy bread made from cassava flour, often served in a bagel-like shape in Paraguay. Also *chipa* means 'rivet, bolt, screw' in Quechua, something for holding things together. The software is available at `http://github.com/alexrudnick/chipa` under the GPL.

- lemmas from surrounding context (three tokens on either side)
- bag of lemmas from the entire sentence
- Brown cluster labels from surrounding context
- bag of Brown cluster labels from the entire sentence

Figure 1: Features used in classification

3.1. System Integration

In order to integrate Chipa into SQUOIA, we added an additional lexical selection stage to the SQUOIA pipeline, occurring after the rule-based disambiguation modules. This new module connects to the Chipa server to request translation suggestions – possibly several per word, ranked by their probability estimates – then looks for words that SQUOIA currently has marked as ambiguous.

For each word with multiple translation possibilities, we consider each of the translations known to SQUOIA and take the one ranked most highly in the results from the classifiers. If there are no such overlapping translations, we take the default entry suggested by SQUOIA's dictionary. Notably, since Chipa and SQUOIA do not share the same lexicon and bitext alignments may be noisy, translations observed in the bitext may be unknown to the SQUOIA system, and lexical entries in the SQUOIA dictionary may not be attested in the training data.

3.2. Learning From Monolingual Data

While in this work, our target language is under-resourced, we have many language resources available for the source language. We would like to use these to make better sense of the input text, giving our classifiers clearer signals for lexical selection in the target language.

One resource for Spanish is its abundant monolingual text. Given large amounts of Spanish-language text, we can use unsupervised methods to discover semantic regularities. In this work we apply Brown clustering (Brown et al., 1992), which has been used successfully in a variety of text classification tasks (Turian et al., 2010) and provides a straightforward mechanism to add features learned from monolingual text.

The Brown clustering algorithm takes as input unannotated text and produces a mapping from word types in that text to clusters, such that words in the same cluster have similar usage patterns according the corpus's bigram statistics. We can then use this mapping from words to clusters in our classifiers, adding an additional annotation for each word that allow the classifiers to find higher-level abstractions than surface-level words or particular lemmas. The desired number of clusters must be set ahead of time, but is a tunable parameter. We use a popular open source implementation of Brown clustering, [3] described by Liang (2005), running on both the Spanish side of our bitext corpus and on the Europarl corpus (Koehn, 2005) for Spanish.

[3] https://github.com/percyliang/brown-cluster

Figure 2 shows some illustrative examples of clusters that we found in the Spanish Europarl corpus. Examining the output of the clustering algorithm, we see some intuitively satisfying results; there are clusters corresponding to the names of many countries, some nouns referring to people, and common transitive verbs. Note that the clustering is unsupervised, and the labels given are not produced by the algorithm.

4. Experiments

Here we report on two basic experimental setups, including an *in-vitro* evaluation of the CL-WSD classifiers themselves and an *in-vivo* experiment in which we evaluate the translations produced by the SQUOIA system with the integrated CL-WSD system.

4.1. Classification Evaluation

To evaluate the classifiers in isolation, we produced a small Spanish-Quechua bitext corpus from a variety of sources, including the Bible, some government documents such as the constitution of Peru and several short folktales and works of fiction. The great majority of this text was the Bible. We used Robert Moore's sentence aligner (Moore, 2002), with the default settings to get sentence-aligned text. Initially there were just over 50 thousand sentences; 28,549 were included after sentence alignment.

During preprocessing, Spanish multi-word expressions identifiable with FreeLing were replaced with special tokens to mark that particular expression, and both the Spanish and Quechua text were lemmatized. We then performed word-level alignments on the remaining sentences with the Berkeley aligner (DeNero and Klein, 2007), resulting in one-to-many alignments such that each Spanish word is aligned to zero or more Quechua words, resulting in a label for every Spanish token.

With this word-aligned bitext, we can then train and evaluate classifiers. We evaluate here classifiers for the 100 most common Spanish lemmas appearing in the aligned corpus. For this test, we performed 10-fold cross-validation for each lemma, retrieving all of the instances of that lemma in the corpus, extracting the appropriate features, training classifiers, then testing on that held-out fold.

We report on two different scenarios for the *in-vitro* setting; in one case, we consider classification problems in which the word in question may be aligned to NULL, and in the other setting, we exclude NULL alignments. While the former case will be relevant for other translation systems, in the architecture of SQUOIA, lexical selection modules may not make the decision to drop a word. In both cases, we show the average classification accuracy across all words and folds, weighted by the size of each test set.

Here we compare the trained classifiers against the "most-frequent sense" (MFS) baseline, which in this setting is the most common translation for a given lemma, as observed in the training data.

We additionally show the effects on classification accuracy of adding features derived from Brown clusters, with clusters extracted from both the Europarl corpus and the Spanish side of our training data. We tried several different settings for the number of clusters, ranging from $C = 100$ to

category	top twenty word types by frequency
countries	francia irlanda alemania grecia italia españa rumanía portugal polonia suecia bulgaria austria finlandia hungría bélgica japón gran_bretaña dinamarca luxemburgo bosnia
more places	kosovo internet bruselas áfrica iraq lisboa chipre afganistán estrasburgo oriente_próximo copenhague asia chechenia gaza oriente_medio birmania londres irlanda_del_norte berlín barcelona
mostly people	hombre periodista jefes_de_estado individuo profesor soldado abogado delincuente demócrata dictador iglesia alumno adolescente perro chico economista gato jurista caballero bebé
infrastructure	infraestructura vehículo buque servicio_público cultivo edificio barco negocio motor avión monopolio planta ruta coche libro aparato tren billete actividad_económica camión
common verbs	pagar comprar vender explotar practicar soportar exportar comer consumir suministrar sacrificar fabricar gobernar comercializar cultivar fumar capturar almacenar curar beber

Figure 2: Some illustrative clusters found by the Brown clustering algorithm on the Spanish Europarl data. These are five out of $C = 1000$ clusters, and were picked and labeled arbitrarily by the authors. The words listed are the top twenty terms from that cluster, by frequency.

system	accuracy				
MFS baseline	54.54				
chipa, only word features	65.43				
	$C = 100$	$C = 200$	$C = 500$	$C = 1000$	$C = 2000$
chipa, +clusters from training bitext	66.71	67.43	68.41	69.00	69.43
chipa, +clusters from europarl	66.60	67.18	67.83	68.25	68.58

Figure 3: Results for the *in-vitro* experiment; classification accuracies over tenfold cross-validation including null-aligned tokens, as percentages.

system	accuracy				
MFS baseline	53.94				
chipa, only word features	68.99				
	$C = 100$	$C = 200$	$C = 500$	$C = 1000$	$C = 2000$
chipa, +clusters from training bitext	71.53	72.62	73.88	74.29	74.78
chipa, +clusters from europarl	71.27	72.08	73.04	73.52	73.83

Figure 4: Classification accuracies over tenfold cross-validation, excluding null-aligned tokens.

$C = 2000$. In all of our experimental settings, the addition of Brown cluster features substantially improved classification accuracy. We note a consistent upward trend in performance as we increase the number of clusters, allowing the clustering algorithm to learn finer-grained distinctions. The training algorithm takes time quadratic in the number of clusters, which becomes prohibitive fairly quickly, so even finer-grained distinctions may be helpful, but will be left to future work. On a modern Linux workstation, clustering Europarl (2M sentences) into 2000 clusters took roughly a day.

The classifiers using clusters extracted from the Spanish side of our bitext consistently outperformed those learned from the Europarl corpus. We had an intuition that the much larger corpus (nearly two million sentences) would help, but the clusters learned in-domain, largely from the Bible, reflect usage distinctions in that domain. Here we are in fact cheating slightly, as information from the complete corpus is used to classify parts of that corpus.

Figures 3 and 4 show summarized results of these first two experiments.

4.2. Translation Evaluation

In order to evaluate the effect of Chipa on lexical selection in a live translation task, we used SQUOIA to translate two Spanish passages for which we had reference Quechua translations. The first is simply a thousand sentences from the Bible; the second is adapted from the Peruvian government's public advocacy website,[4] which is bilingual and presumably contains native-quality Quechua. We collected and hand-aligned thirty-five sentences from this site.

Having prepared sentence-aligned and segmented bitexts for the evaluation, we then translated the Spanish side with SQUOIA, with various CL-WSD settings to produce Quechua text. In comparing the output Quechua with the reference translations, BLEU scores were quite low. The output often contained no 4-grams that matched with the reference translations, resulting in a geometric mean of 0. So here we report on the unigram-BLEU scores, which reflect some small improvements in lexical choice. See Figure 5 for the numerical results.

On the web test set, unfortunately very few of the Spanish

[4] *Defensoría del Pueblo*, `http://www.defensoria.gob.pe/quechua.php`

system	web test set	bible test set
squoia without CL-WSD	28.1	24.2
squoia+chipa, only word features	28.1	24.5
squoia+chipa, +europarl clusters	28.1	24.5
squoia+chipa, +bible clusters	28.1	24.5

Figure 5: BLEU-1 scores (modified unigram precision) for the various CL-WSD settings of SQUOIA on the two different Spanish-Quechua test sets.

words used were both considered ambiguous by SQUOIA's lexicon and attested in our training corpus. Enabling Chipa during translation, classifiers are only called on six of the thirty-five sentences, and then the classifiers only disagree with the default entry from the lexicon in one case.

We do see a slight improvement in lexical selection when enabling Chipa on the Bible test set; the three feature settings listed actually all produce different translation output, but they are of equal quality. Here the in-domain training data allowed the classifiers to be used more often; 736 of the thousand sentences were influenced by the classifiers in this test set.

5. Related Work

Framing the resolution of lexical ambiguities in machine translation as an explicit classification task has a long history, dating back at least to early SMT work at IBM (Brown et al., 1991). More recently, Carpuat and Wu have shown how to use classifiers to improve modern phrase-based SMT systems (Carpuat and Wu, 2007). CL-WSD has received enough attention to warrant shared tasks at recent SemEval workshops; the most recent running of the task is described by Lefever and Hoste (2013). In this task, participants are asked to translate twenty different polysemous English nouns into five different European languages, in a variety of contexts.

Lefever *et al.*, in work on the ParaSense system (2011), produced top results for this task with classifiers trained on local contextual features, with the addition of a bag-of-words model of the translation of the complete source sentence into other (neither the source nor the target) languages. At training time, the foreign bag-of-words features for a sentence are extracted from available parallel corpora, but at testing time, they must be estimated with a third-party MT system, as they are not known a priori. This work has not yet, to our knowledge, been integrated into an MT system on its own.

In our earlier work, we prototyped a system that addresses some of the issues with ParaSense, requiring more modest software infrastructure for feature extraction while still allowing CL-WSD systems to make use of several mutually parallel bitexts that share a source language (Rudnick et al., 2013). We have also done some previous work on CL-WSD for translating into indigenous American languages; an earlier version of Chipa, for Spanish-Guarani, made use of sequence models to jointly predict all of the translations for a sentence at once (Rudnick and Gasser, 2013).

Francis Tyers, in his dissertation work (2013), provides an overview of lexical selection systems and describes methods for learning lexical selection rules based on available parallel corpora. These rules make reference to the lexical items and parts of speech surrounding the word to be translated. Once learned, these rules are intended to be understandable and modifiable by human language experts. For practical use in the Apertium machine translation system, they are compiled to finite-state transducers.

Rios and Göhring (2013) describe earlier work on extending the SQUOIA MT system with machine learning modules. They used classifiers to predict the target forms of verbs in cases where the system's hand-crafted rules cannot make a decision based on the current context.

6. Conclusions and Future Work

We have described the Chipa CL-WSD system and its integration into SQUOIA, a machine translation system for Spanish-Quechua. Until this work, SQUOIA's lexical choices were based on a small number of hand-written lexical selection rules, or the default entries in a bilingual dictionary.

We have provided a means by which the system can make some use of the available training data, both bilingual and monolingual, with very few changes to SQUOIA itself. We have also shown how Brown clusters, either when learned from a large out-of-domain corpus or from a smaller in-domain corpus, provide useful features for a CL-WSD task, substantially improving classification accuracy.

In order make better use of the suggestions from the CL-WSD module, we may need to expand the lexicon used by the translation system, so that mismatches between the vocabulary of the available bitext, the translation system itself, and the input source text do not hamper our efforts at improved lexical selection. Finding more and larger sources of bitext for this language pair would of course help immensely.

We would like to learn from the large amount of monolingual Spanish text available; while the Europarl corpus is nontrivial, there are much larger sources of Spanish text, such as the Spanish-language Wikipedia. We plan to apply more clustering approaches and other word-sense discrimination techniques to these resources, which will hopefully further improve CL-WSD across broader domains.

Better feature engineering outside of unsupervised clusters may also be useful. In the future we we will extract features from the already-available POS tags and the syntactic structure of the input sentence.

We also plan to apply the Chipa system to other machine translation systems and other language pairs, especially Spanish-Guarani, another important language pair for South America.

7. References

Academia Mayor de La Lengua Quechua. (2005). *Diccionario: Quechua - Español - Quechua, Qheswa - Español - Qheswa: Simi Taqe, 2da ed.* Cusco, Perú.

Agirre, E. and Edmonds, P. G. (2006). *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science+ Business Media.

Alegria, I., de Ilarraza, A. D., Labaka, G., Lersundi, M., Mayor, A., Sarasola, K., Forcada, M. L., Rojas, S. O., and Padró, L. (2005). An Open Architecture for Transfer-based Machine Translation between Spanish and Basque. In *Workshop on Open-source machine translation at Machine Translation Summit X*.

Attardi, G., Dell'Orletta, F., Simi, M., Chanev, A., and Ciaramita, M. (2007). Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1112–1118.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1991). Word-Sense Disambiguation Using Statistical Methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Carpuat, M. and Wu, D. (2007). How Phrase Sense Disambiguation Outperforms Word Sense Disambiguation for Statistical Machine Translation. In *11th Conference on Theoretical and Methodological Issues in Machine Translation*.

DeNero, J. and Klein, D. (2007). Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, June.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of The Tenth Machine Translation Summit*.

Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lefever, E. and Hoste, V. (2013). SemEval-2013 Task 10: Cross-Lingual Word Sense Disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*.

Lefever, E., Hoste, V., and De Cock, M. (2011). ParaSense or How to Use Parallel Corpora for Word Sense Disambiguation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Liang, P. (2005). Semi-supervised learning for natural language. Master's thesis, MIT.

Mayor, A., Alegria, I., Díaz de Ilarraza, A., Labaka, G., Lersundi, M., and Sarasola, K. (2011). Matxin, an open-source rule-based machine translation system for basque. *Machine Translation*, 25(1):53–82.

Moore, R. C. (2002). Fast and accurate sentence alignment of bilingual corpora. In *AMTA*, pages 135–144.

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

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

Rios, A., Göhring, A., and Volk, M. (2009). A quechua–spanish parallel treebank. In *Proceedings of 7th Workshop on Treebanks and Linguistic Theories (TLT-7), Groningen*.

Rios Gonzales, A. and Göhring, A. (2013). Machine Learning Disambiguation of Quechua Verb Morphology. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, Sofia, Bulgaria.

Rudnick, A. and Gasser, M. (2013). Lexical Selection for Hybrid MT with Sequence Labeling. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, pages 102–108, Sofia, Bulgaria.

Rudnick, A., Liu, C., and Gasser, M. (2013). HLTDI: CL-WSD Using Markov Random Fields for SemEval-2013 Task 10. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*.

Turian, J., Ratinov, L.-A., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden.

Tyers, F. M. (2013). *Feasible lexical selection for rule-based machine translation*. Ph.D. thesis, Departament de Llenguatges i Sistemes Infomàtics, Universitat d'Alacant.