The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014

EDITORIAL BOARD

Published twice a year by Charles University in Prague

Editorial office and subscription inquiries:
ÚFAL MFF UK, Malostranské náměstí 25, 118 00, Prague 1, Czech Republic
E-mail: pbml@ufal.mff.cuni.cz

ISSN 0032-6585

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014

CONTENTS

Articles

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 5–16

Qualitative: Open source Python tool for Quality Estimation over multiple Machine Translation outputs

Eleftherios Avramidis, Lukas Poustka, Sven Schmeier

German Research Center for Artificial Intelligence (DFKI Berlin)

Abstract

"Qualitative" is a python toolkit for ranking and selection of sentence-level output by different MT systems using Quality Estimation. The toolkit implements a basic pipeline for annotating the given sentences with black-box features. Consequently, it applies a machine learning mechanism in order to rank data based on models pre-trained on human preferences. The preprocessing pipeline includes support for language models, PCFG parsing, language checking tools and various other pre-processors and feature generators. The code follows the principles of object-oriented programming to allow modularity and extensibility. The tool can operate by processing both batch-files and single sentences. An XML-RPC interface is provided for hooking up with web-services and a graphical animated web-based interface demonstrates its potential on-line use.

1. Introduction

Having been a topic of research for many years, Machine Translation (MT) has recently reached a wide range of applications for big audiences. Methods and analyses produced in academic labs have been adopted by the industry and transferred to products and services with numerous use-cases. This has increased the interest for the field and generously empowered a broad spectrum of research activities and further development.

In this long history of conveying MT research into everyday use, the software developed by the relevant research community has been of high value. Through the open source implementation of majorly important MT and natural language processing tools (Koehn et al., 2007; Federico et al., 2008), researchers had the opportunity

 Corresponding author: eleftherios.avramidis@dfki.de
Cite as: Eleftherios Avramidis, Lukas Poustka, Sven Schmeier. Qualitative: Open source Python tool for Quality Estimation over multiple Machine Translation outputs. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 5–16. doi: 10.2478/pralin-2014-0009.

to test and expand proposed methods and easily introduce new ideas. Additionally, the industry found pretty strong software engineering prototypes that were not too far from user-oriented programs and many of them were directly plugged into user interfaces.

Adopting this line of MT-oriented software development, we provide a Free Software implementation in the direction of Quality Estimation (QE). This can be directly used for further research/applications on MT output ranking, system selection, hybrid machine translation etc. Moreover, it provides the basics for further community development on QE, aiming to gather a wide range of contributions.

The following sections contain a comparison of our software with already existing open source tools (section 2), a description of the basic functionality (section 3), an explanation of the methods used in the background (section 4), an introductory guide for further development (section 5) and the plans for further work (section 6).

2. Previous work

The first collaborative work for development on this field was done in the frame of the WS'03 Summer Workshop at the John Hopkins University on *Confidence Estimation of MT* (Blatz et al., 2004), but to our knowledge no application or code has been publically available, as a result of this effort. Additionally, relevant contributions introduced several open-source tools offering MT evaluation (e.g. METEOR (Banerjee and Lavie, 2005)), Hjerson and Addicter (Berka et al., 2012)), whereas a good amount of such metrics and scripts were gathered in Asiya (Giménez and Marquez, 2010). All this work is based on comparing the produced translations with reference translations, which generally falls out of the scope of QE.

A major contribution directly to the field of QE has been done by the QuEst tool (Specia et al., 2013), which included an implementation of various features by numerous contributors. Whereas there is serious overlapping of QuEst with our submission, there are notable differences. In contrast to the regression-based orientation of QuEst, our software is aimed to sentence-level ranking and selection of MT-output, by implementing comparative Quality Estimation. Additionally, not the same quality features have been implemented in both toolkits, whereas Python as a dynamic language allows provides more flexible data structures and architecture. Nevertheless, our software includes a QuEst wrapper for conformity with WMT baselines.

3. Basic use

The out-of-the box functionality of the software is based on a use-case, where one source sentence is translated by several MT systems. The software therefore analyzes properties of all translations and suggests the proper *ranking* (ordering), trying to predict human preference. This can be used for ranking system outputs or combining several systems on the sentence level.

3.1. Installation

The toolkit is implemented in Python 2.7 and has been developed and tested for operation in a Linux operating system. The code has to be downloaded[1] and the Python path needs to be set to the `/src` directory, where all python scripts, modules and packages reside. Much of the functionality is provided by several publically available Python extensions, which need to be installed prior to any execution. All extensions required for the out-of-the-box functionality are provided by the Python *pip* package management system, so it is enough to run the respective `pip install` commands for the packages detailed in `INSTALL.txt`, These installations can easily take place on the user's home folder, without requiring root access (e.g. in experiment server environments).

The toolkit interacts with several java applications whose 'jar' and class files have to be placed in the directory `/lib`. An installation script that automatically downloads all required java dependencies is provided. Additionally, one needs to execute externally the LM server by Nitin Madnani.[2]

3.2. Resources and Configuration

The quality features for the given sentences are generated within a pipeline of NLP analysis tools. Many of these tools require specific resources to be acquired prior to the execution of the program. In particular, for the out-of-the-box installation and pre-trained models, one needs a language model, a PCFG grammar and a truecaser model for the source and target languages, which are also provided by an installation script through our servers.

All file locations and several other parameters (e.g. the translation language direction) can be specified in one or more complementary configuration files. Sample configuration files are provided in `/cfg/autoranking` and can be modified accordingly to fit the user's specific installation. The configuration files may also include references to many language-specific resources; the program will only use the ones which are relevant to the chosen languages.

The reason for allowing many configuration files is that one may want to split the configuration parameters depending on the environment, i.e. some settings may be generic and applicable to all computational servers, whereas some others may change from machine to machine.

---

[1] `http://www.dfki.de/~elav01/software/qualitative`

[2] `http://desilinguist.org` At the time that this paper is submitted, the Language Model scores are provided by using LM server, which wraps around SRILM (Stolcke, 2002) and needs to be compiled and executed separately. It is in our plans to remove this dependency and include KenLM (Heafield, 2011)

3.3. Execution

There are several ways the program can be executed. All relevant scripts can be found in the directory `/src/app/autoranking`. In particular, the basic functionality is provided as following:

- **Command-line interaction** (`application.py`): This script allows the user to interact with the sentence selection on the commandline. A configuration file and a pre-trained selection model need to be passed as parameters. Then the user can sequentially type the source sentence and the respective translations one by one. The purpose of this script is mainly to verify that all installation and settings are correct.
- **Batch decoding** (`decode_batch.py`): This script serves for cases where multiple sentences with their respective translations need to be analyzed in a row (e.g. for translating full documents). Apart from the configuration file and the trained classifier, this script accepts the decoding data in an XML file.
- **XML-RPC interface** (`xmlrpcserver.py`): This instantiates and XML-RPC awaiting translation requests for one sentence at a time. The server responds to the command `rank`, having as parameters the source and any number of respective translations. It is useful for binding to web-applications.

In order to assess the installation process we provide pre-trained model for German-English.

3.4. Demonstration server

As a demonstration for the use of the toolkit in a web service, an additional piece of software is provided. It is a web-interface[3] implemented in PHP which allows the user to enter source text to be translated. Consequently, it communicates with external translation services (e.g. Moses server, Google Translate API, Lucy RBMT), and fetches the translations. Finally, the produced translations are given to our ranking/selection mechanism and the result is visualised graphically.

The demonstration server is provided as an example. It is not included in the architecture of the rest of the toolkit and therefore is distributed as a separate package. Users have to modify the script, in order to parametrise the necessary URL addresses and configuration.

4. Under the hood

The core of the program is based on comparative Quality Estimation. The given sentence is first processed within a pipeline of modules. These modules perform text pre-processing and various NLP analyses to generate features that indicate the quality

[3]The demo web interface can be accessed at `http://www.qt21.eu/demonstrator`

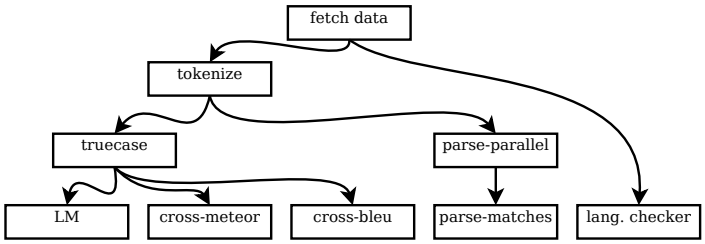


Figure 1. Sample pipeline of feature generators for one of the most successful feature sets.

of the translation. The generated features are consequently fed to a machine learning algorithm, which employs a statistical model for putting the translations in an order of preference. This statistical model has been previously trained on human-annotated data.

In principle, there can be various combinations of features and machine learning algorithms, depending on what performs best given various properties of the quality estimation task. The optimal combination, which performs similarly to human decisions, is subject of constant research in the field. The provided vanilla implementation and pre-trained model follow one of the most successful experiments on German-English, which includes the feature set #24 with Logistic Regression as described in Avramidis (2013a).

4.1. Generation of features

As explained above, the generation of features is a crucial step for acquiring quality hints regarding the processed sentence. For this purpose, the toolkit provides a set of modules, thereof called *feature generators*. A sample pipeline can be seen in Figure 1.

- **Language model** (LM): It sends a query to the language model in order to acquire LM probabilities for unigrams, bigrams, trigrams etc., and also detects and counts words unknown to the language model. It also saves the sentence position (index) of the unknown words and the n-grams with the lowest and highest probability. Features also include the average and the standard deviation of the n-grams probabilities and the sentence positions. The queries to the LM can be sent via XML-RPC to an external LM server, or to call the respective functions from an imported LM library (e.g. KenLM).
- **PCFG parsing**: It loads a language-specific PCFG grammar and handles the parsing of source and target sentences by PCFG parsing. It extracts the overall log-likelihood, the best parse confidence and the count (k) of k-best parse trees generated. The best parse is included as string meta-data, so that following fea-

ture generators can re-use it for producing their features. One of them counts how many times each tree node label appears in the parse and calculate their respective sentences position statistics. Another performs naïve source-to-target tree-label matching, i.e. calculates the ratio of VPs on the source with the respective VPs on the target. The current implementation supports the Berkeley Parser (Heafield, 2011) via either as included library or as an external XML-RPC server. There are pre-trained grammars for English, German, Spanish (Taulé et al., 2008), French, Russian, Chinese and Arabic freely available.

- **Cross-target BLEU and METEOR**: It encapsulates the calculation of well-known reference-aware n-gram-based metrics. For the scenario of multiple alternative translations by different systems, each particular system receives as a feature its own metric score, as if the other competitive systems were used as reference translations. For certain scenarios, this may indicate cases when a system output is very different than the majority of the other competitive system outputs. Here, smoothed sentence level BLEU (Papineni et al., 2002) and all METEOR components (precision, recall, fragmentation penalty and overall weighed score) (Lavie and Agarwal, 2007) are supported.
- **Language correction**: This investigates the usability of language-correction software. Such software is based on hand-written rules which detect grammatical, syntactic and other expresional inconsistencies on monolingual text usually while being written in word processors; the errors are automatically flagged and suggestions are given to authors. We use the count of each error type in every sentence as a separate feature, a tactic that unfortunately produces rather sparse features. A feature generator wraps around the open source Language Tool library (Naber, 2003; Miłkowski, 2012), whereas remote querying towards the optional proprietary software Acrolinx IQ (Siegel, 2011) is also supported via the SUDS protocol.
- **IBM1 probabilities**: This feature generator supports reading an IBM-1 model and includes the produced sentence probability as an additional feature.
- **Length features** include simple counting of the number of tokens, characters and the average number of characters per word for each sentence.
- **Ratios and differences**: A last step calculates differences and ratios between every source and its respective translation feature, if available. Defining explicit features for known relations between features may be useful for some ML algorithms.

The code includes the default normalisation, tokenisation and truecasing scripts of Moses (Koehn et al., 2007). Additional tokenisation and other pre-processing actions, when needed, are done via NLTK (Loper and Bird, 2002). Also, some functions from automatic error extraction of Hjerson (Popović, 2011) are included.

4.2. Machine Learning

The ranking algorithm is based on binary classifier decisions, which are recombined up into a full ranking (Usunier et al., 2009; Avramidis, 2012). The decision is every time taken on a sentence level, given the numeric features generated for this particular sentence and translation. The final algorithm has no access or use for the given/translated text. Although the pre-trained models use logistic regression (Hosmer, 1989) as a pairwise classifier, the interface allows many other learning methods to be employed and stored, such as SVM (Joachims, 2006), Naive Bayes, k-Nearest neighbours (Coomans and Massart, 1982) and Decision Trees (Quinlan, 1986).

4.3. Training

Training of new models takes places in two stages, *annotation* and *learning*. First, the training data need to be processed by the feature generators pipeline using the batch annotation script[4] and a given configuration script. Implementing the interface provided by the Ruffus library (Goodstadt, 2010) allows for parallelisation of intermediate steps into a number of CPU cores, whereas it keeps track of finished and unfinished steps so that they can be resumed if something crashes. Heavy and slow tasks, such as parsing, are also parallelised after being split into multiple parts.

On the second phase, the learning script[5] trains and evaluates the machine learning algorithms given the features generated previously at the annotation phase. Both learning algorithms and feature sets can be defined in a configuration file. The learning pipeline is organised through a modified version of the Python Experiment Suite (Rückstieß and Schmidhuber, 2011), so intermediate steps are saved on the disk for resuming or further use, as for example the pickled classifier models.

Machine learning algorithms are primarily provided by Orange (Demšar et al., 2004) and optionally by Scikit-learn (Pedregosa et al., 2011). The entire set of RankEval scripts (Avramidis, 2013b) are included as part of the software. Many of the evaluation functions and calculations are based on NumPy and SciPy (Oliphant, 2007).

5. Development

The architecture of the program has been designed to allow for further development. The code is open and collaborative efforts are centralised within a Git repository.[6]. The development has been divided in several python packages. Each package serves a different function, so that the code can be modural and re-usable.

[4] `/src/app/autoranking/annotate_batch.py`

[5] `/src/app/autoranking/suite.py`

[6] Please check `http://www.dfki.de/~elav01/software/qualitative` for details on how to clone the repository and commit

The code has been documented based on inline comments following the EpyDoc standard, and therefore an automatically generated API documentation is provided for the vast majority of functions and classes. The more important classes and functions are detailed in the following sections.

5.1. Understanding the data structures

The data structures used in all parts of the software are contained in the `sentence` package. The basic structures are:

- The `SimpleSentence` is the basic data structure. It is a class which wraps the original sentence text as a string. Additionally, it contains a dictionary of "attributes". These can be features and meta-data provided by the original data; they are further augmented by the annotation process (see feature generators) and they are the ones who are important for the machine learning algorithms, which also put their results as attributes.
- The `ParallelSentence` is a class that represents the basic unit of a parallel corpus. It contains one source sentence, a list of target sentences and optionally a reference, All encapsulated sentences are an instance of `SimpleSentence`. The `ParallelSentence` also includes its own attribute dictionary, with the sentence id, the source and target language asclasses the most common attributes.
- The `DataSet` is an iterable class which stands one level above, as it encapsulates a list of parallel sentences and several convenience functions.

Any development effort should use the data structures, which allow for optimal flow of data between the various modules. In this python package, one can find more classes which extend the basic data structures. These extensions have been developed to support breaking the parallel sentences into a set of pairwise sentence comparisons.

5.2. Reading and writing files

In order to facilitate processing data for sentence-level ranking, external data processing is based on a XML format called "JCML" for *Judged Corpus Markup Language*. In contrast to line-separated simple-text formats used for regression-based QE, this format allows for a variable number of target sentences per source[7], whereas all features are aggregated in the same file. In the package `support.jcml` we provide several utility scripts (e.g. for joining and splitting) and also a utility for converting from multiple plain text files to JCML.

Reading and writing external data is taken care through the classes in the package `dataprocessor`. The modules contained in the package allow for reading and writing using several alternative python libraries, e.g. *minidom* (batch all-in-memory),

[7] The ranking-based human evaluation tasks of WMT provides 2-12 system outputs per source sentence. For this reason JCML was used for the QE ranking task at WMT13

SAX and *CElementTree* (incremental disk writing/reading). The most commonly-used reader is from `ce.cejcml`, whereas the most common writer is `IncrementalJcml` from `sax.saxps2jcml`.

5.3. Adding new features

The classes responsible for generating features reside in the `featuregenerator` package. One can add features by developing a new feature generator. The required steps are:

- create a new module with a new class that extends `FeatureGenerator` (from `featuregenerator`). The initialisation function should load necessary resources as named arguments. If more modules are required, place them in a new package.
- override the unimplemented function `get_features_tgt`, perform the required analysis of each target sentence and return a dictionary containing the resulting feature names and values. This function will be automatically repeated for all target sentences and can also process the source sentence.
- optionally override the unimplemented function `get_features_src` if you want to provide features that refer only to the source sentence. Similarly prefixed functions can be overridden for attaching features as attributes to other parts of the parallel sentence.
- optionally override the functions `add_features_tgt` and similarly prefixed function if you also want to modify the string of the processed sentences (e.g. for pre-processing, tokenisation etc.).
- add the new feature generator in the annotation process of the relevant application. An initialised instance of the class should be added in a list with all other feature generators that are executed. The order of the feature generators in the list matters, particularly when some generators require pre-processing or meta-data from previously executed generators. If possible, parameters should be loaded from a configuration file.

It is also possible to encapsulate tools and libraries written in Java through the Py4J library, following the example of the `BerkeleyParserSocket` class.

5.4. New QE applications and machine learning algorithms

The provided code includes implementation for the ranking mechanism by using pairwise classifiers. Since there is infrastructure for feature generation and machine learning, other experiments and applications can be easily developed. Additional **QE applications** can be added as separate packages in `apps` by following the example of the `autoranking` package. These are mainly executable scripts for annotation and training (see section 4.3), or other functions (test module, commandline app or XML-RPC server).

As mentioned, this package already provides support for several functions of Orange and Scikit-learn. Further **machine learning algorithms** can be included in `ml.lib` by implementing the abstract functions of the classes in the package `ml`.

6. Further work

Whereas the provided code contains a fully functional implementation and a modular architecture, several parts are subject of further improvement. We are currently working on improving the architecture, e.g. to provide abstract classes for machine learning methods or better templates for new "apps". Additionally, a constant goal are more feature implementations, to cover at least the so-called *baseline* features. Readers of this paper are advised to check the latest version of documentation and architecture in the official web-page.

Acknowledgements

This work has received support by the EC's FP7 (FP7/2007-2013) under grant agreement number 610516: "QTLeap: Quality Translation by Deep Language Engineering Approaches". Early stages have been developed with the support of the projects TaraXŰ and QT-Launchpad. Many thanks to: Slav Petrov for modifying the Berkeley Parser in order to allow modification of parsing parameters; Hieu Hoang, Philipp Koehn, Maja Popović, Josh Schroeder and David Vilar as parts of their open source code have been included in some of our scripts; Aljoscha Burchardt and Prof. Hans Uszkoreit for the support.

Bibliography

Avramidis, Eleftherios. Comparative Quality Estimation: Automatic Sentence-Level Ranking of Multiple Machine Translation Outputs. In *Proceedings of 24th International Conference on Computational Linguistics*, pages 115–132, Mumbai, India, Dec. 2012. The COLING 2012 Organizing Committee.

Avramidis, Eleftherios. Sentence-level ranking with quality estimation. *Machine Translation (MT)*, 28(Special issue on Quality Estimation):1–20, 2013a.

Avramidis, Eleftherios. RankEval: Open Tool for Evaluation of Machine-Learned Ranking. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 100:63–72, 2013b. doi: 10.2478/pralin-2013-0012.

Banerjee, Somnath and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, 2005.

Berka, Jan, Ondřej Bojar, Mark Fishel, Maja Popović, and Daniel Zeman. Automatic MT Error Analysis: Hjerson Helping Addicter. In *8th International Conference on Language Resources and Evaluation*, pages 2158–2163, 2012. ISBN 978-2-9517408-7-7.

Blatz, John, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence Estimation for Machine Translation. In Rollins, M., editor, *Mental Imagery*. Yale University Press, 2004.

Coomans, D. and D.L. Massart. Alternative k-nearest neighbour rules in supervised pattern recognition. *Analytica Chimica Acta*, (138):15–27, Jan. 1982. ISSN 00032670.

Demšar, Janez, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From Experimental Machine Learning to Interactive Data Mining. In *Principles of Data Mining and Knowledge Discovery*, pages 537–539, 2004.

Federico, Marcello, Nicola Bertoldi, and Mauro Cettolo. IRSTLM: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621. ISCA, 2008.

Giménez, Jesús and Lluís Marquez. Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. *The Prague Bulletin of Mathematical Linguistics*, 94:77–86, 2010. doi: 10.2478/v10108-010-0022-6.

Goodstadt, Leo. Ruffus: a lightweight Python library for computational pipelines. *Bioinformatics*, 26(21):2778–2779, Nov. 2010. ISSN 1367-4803.

Heafield, Kenneth. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Hosmer, David. *Applied logistic regression*. Wiley, New York [u.a.], 8th edition, 1989. ISBN 9780471615538.

Joachims, Thorsten. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006. ISBN 1595933395.

Koehn, Philipp, 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. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180, Prague, Czech Republic, June 2007.

Lavie, Alon and Abhaya Agarwal. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Loper, Edward and Steven Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Miłkowski, Marcin. *Translation Quality Checking in LanguageTool*, pages 213–223. Corpus Data across Languages and Disciplines. Peter Lang, Frankfurt am Main, Berlin, Bern, Bruxelles, New York, Oxford, Wien, 2012.

Naber, Daniel. A rule-based style and grammar checker. Technical report, Bielefeld University, Bielefeld, Germany, 2003.

Oliphant, Travis E. SciPy: Open source scientific tools for Python, 2007.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

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

Popović, Maja. Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 96(1):59–68, 2011. doi: 10.2478/v10108-011-0011-4.

Quinlan, J. R. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, Mar. 1986. ISSN 0885-6125.

Rückstieß, Thomas and Jürgen Schmidhuber. Python Experiment Suite Implementation. *The Python Papers Source Codes*, 2:4, 2011.

Siegel, Melanie. Autorenunterstützung für die Maschinelle Übersetzung. In *Multilingual Resources and Multilingual Applications: Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, Hamburg, 2011.

Specia, Lucia, Kashif Shah, José Guilherme Camargo de Souza, and Trevor Cohn. QuEst - A translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.

Stolcke, Andreas. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904. ISCA, Sept. 2002.

Taulé, Mariona, Antònia Martí, and Marta Recasens. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). ISBN 2-9517408-4-0.

Usunier, Nicolas, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th Annual International Conference on Machine Learning ICML 2009 Montreal Quebec Canada June 1418 2009*, pages 1057—-1064. ACM, 2009.

Address for correspondence:
Eleftherios Avramidis
eleftherios.avramidis@dfki.de
German Research Center for Artificial Intelligence (DFKI GmbH)
Language Technology Lab
Alt Moabit 91c
10559 Berlin, Germany

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 17–26

A Fast and Simple Online Synchronous Context Free Grammar Extractor

Paul Baltescu, Phil Blunsom

University of Oxford, Department of Computer Science

Abstract

Hierarchical phrase-based machine translation systems rely on the synchronous context free grammar formalism to learn and use translation rules containing gaps. The grammars learned by such systems become unmanageably large even for medium sized parallel corpora. The traditional approach of preprocessing the training data and loading all possible translation rules into memory does not scale well for hierarchical phrase-based systems. Online grammar extractors address this problem by constructing memory efficient data structures on top of the source side of the parallel data (often based on suffix arrays), which are used to efficiently match phrases in the corpus and to extract translation rules on the fly during decoding. This paper describes an open source implementation of an online synchronous context free grammar extractor. Our approach builds on the work of Lopez (2008a) and introduces a new technique for extending the lists of phrase matches for phrases containing gaps that reduces the extraction time by a factor of 4. Our extractor is available as part of the `cdec` toolkit[1] (Dyer et al., 2010).

1. Introduction

Grammar extraction is the part of a machine translation pipeline responsible for finding the set of applicable translation rules in a word-aligned parallel corpus. Every time a machine translation system receives a sentence as input, the extractor is queried for the set of translation rules that match subphrases of the given sentence. The overall translation time depends on the extractor's ability to efficiently identify these rules. This paper introduces a fast and simple grammar extractor for hierarchical phrase based translation systems.

[1] Our code is available here: `https://github.com/redpony/cdec/tree/master/extractor`.

 Corresponding author: paul.baltescu@cs.ox.ac.uk
Cite as: Paul Baltescu, Phil Blunsom. A Fast and Simple Online Synchronous Context Free Grammar Extractor. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 17–26. doi: 10.2478/pralin-2014-0010.

The traditional approach to grammar extraction is achieved with the help of phrase tables, dictionary-like data structures that map all the source phrases in the training corpus to their target side candidates. Phrase tables are generated in a preprocessing step, by iterating over each sentence in the word-aligned parallel corpus and extracting all phrase pairs up to a fixed width, such that a translation rule contains a word only if it also contains all the words aligned to it (Och and Ney, 2004). Phrase tables have been designed with phrase-based systems in mind (Koehn et al., 2003; Och and Ney, 2004), where the number of extractable phrase pairs is linear in the phrase width parameter. Even so, loading all the translation rules into memory can be problematic for large corpora or in memory constrained environments (mobile devices, commodity machines, etc.).

Hierarchical phrase-based translation systems (Chiang, 2007) learn and use translation rules containing gaps. For such systems, the number of extractable translation rules is exponential in the phrase width parameter. As a result, the grammars learned by hierarchical phrase-based systems are too large to fit in memory for almost all relevant setups. A naive solution is to filter the phrase tables and remove all translation rules that are not applicable for a given test set, but this approach does not scale to unseen sentences which are to be expected by any machine translation application.

Several memory efficient alternatives to phrase tables have been proposed. Zens and Ney (2007) store phrase tables on disk organized in a prefix tree data structure for efficient read access. Callison-Burch et al. (2005) and Zhang and Vogel (2005) introduce a phrase extraction technique based on suffix arrays which extracts translation rules on the fly during decoding. Lopez (2007) shows how online extractors based on suffix arrays can be extended to handle phrases with gaps. These two approaches have comparable lookup times despite the fact that the former has a better asymptotic complexity (constant vs. logarithmic) because slower disk reads are involved. In most scenarios, the suffix array approach is preferred (e.g. Schwartz and Callison-Burch (2010)) because it yields several benefits. The phrase width limitation for translation rules is no longer required as it has no effect on the memory footprint of the precomputed data structures. Also, less time is spent when tuning translation models, as the precomputed data structures need not be constructed again when new scoring features are added.

The remainder of this paper describes our suffix array based grammar extractor for hierarchical phrase-based systems. Section 2 reviews how suffix arrays are used for contiguous phrase extraction (Lopez, 2008a). Section 3 introduces our new technique for extracting phrases with gaps. Section 4 briefly covers the intended usage for our tool and discusses other implementation specific details which might make our tool appealing to the research community. Section 5 concludes the paper with a set of experiments demonstrating the benefits of the novel technique introduced in this paper and other speed gains obtained as a result of careful implementation.

2. Grammar extraction for contiguous phrases

A suffix array (Manber and Myers, 1990) is a memory efficient data structure which can be used to efficiently locate all the occurrences of a pattern, given as part of a query, in some larger string (referred to as *text* in the string matching literature, e.g. Gusfield (1997)). A suffix array is simply the list of suffixes in the text string sorted in lexicographical order. A suffix is encoded by its starting position and the overall size of the suffix array is linear in the size of text string. A crucial property of suffix arrays is that all suffixes starting with a given prefix form a compact interval within the suffix array.

Suffix arrays are well suited to solve the central problem of contiguous phrase extraction: efficiently matching phrases against the source side of the parallel corpus. Once all the occurrences of a certain phrase are found, candidate translation rules are extracted from a subsample of phrase matches. The rule extraction algorithm (Och and Ney, 2004) is linear in the size of the phrase pattern and adds little overhead to the phrase matching step.

Before a suffix array can be applied to the phrase matching problem, the source side of the parallel corpus is preprocessed as follows: first, words are replaced with numerical ids and then all sentences are concatenated together into a single array. The suffix array is constructed from this array. In our implementation, we use a memory efficient suffix array construction algorithm proposed by Larsson and Sadakane (2007) having $O(N \log N)$ time complexity. The memory requirements of the suffix array are linear in the size of the training data.

The algorithm for finding the occurrences of a phrase in the parallel corpus uses binary search to locate the interval of suffixes starting with that phrase pattern in the suffix array. Let $w_1, w_2, \ldots, w_K$ be the phrase pattern. Since a suffix array is a sorted list of suffixes, we can binary search the interval of suffixes starting with w_1. This contiguous subset of suffix indices continues to be lexicographically sorted and binary search may be used again to find the subinterval of suffixes starting with w_1, w_2. However, all suffixes in this interval are known to start with w_1, so it is sufficient to base all comparisons on only the second word in the suffix. The algorithm is repeated until the whole pattern is matched successfully or until the suffix interval becomes empty, implying that the phrase does not exist in the training data. The complexity of the phrase matching algorithm is $O(K \log N)$. We note that $w_1, \ldots, w_{K-1}$ is a subphrase of the input sentence as well and the extractor applies the phrase matching algorithm for $w_1, \ldots, w_{K-1}$ as part of a separate query. Matching $w_1, \ldots, w_{K-1}$ executes the first $K-1$ steps of the phrase matching algorithm for $w_1, \ldots, w_K$. Therefore, the complexity of the matching algorithm can be reduced to $O(\log N)$ per phrase, by caching the suffix array interval found when searching for $w_1, \ldots, w_{K-1}$ and only executing the last step of the algorithm for $w_1, \ldots, w_K$.

Let M be the length of a sentence received as input by the decoder. If the decoder explores the complete set of contiguous subphrases of the input sentence, the suffix

array is queried $O(M^2)$ times. We make two trivial observations to further optimize the extractor by avoiding redundant queries. These optimizations do not lead to major speed-ups for contiguous phrase extraction, but are important for laying the foundations of the extraction algorithm for phrases containing gaps. First, we note that if a certain subphrase of the input sentence does not occur in the training corpus, any phrase spanning this subphrase will not occur in the corpus as well. Second, phrases may occur more than once in a test sentence, but all such repeating occurrences share the same matches in the training corpus. We add a caching layer on top of the suffix array to store the set of phrase matches for each queried phrase. Before applying the pattern matching algorithm for a phrase $w_1, \ldots, w_K$, we verify if the cache does not already contain the result for $w_1, \ldots, w_K$ and check if the search for $w_1, \ldots, w_{K-1}$ and $w_2, \ldots, w_K$ returned any results. The caching layer is implemented as a prefix tree with suffix links and constructed in a breadth first manner so that shorter phrases are processed before longer ones (Lopez, 2008a).

3. Grammar extraction for phrases with gaps

Synchronous context free grammars are the underlying formalism which enable hierarchical translation systems to use translation rules containing gaps. For a detailed introduction to synchronous context free grammars in machine translation see Lopez (2008b). In this section, we present an algorithm for extracting synchronous context free rules from a parallel corpus, which requires us to adapt the phrase extraction algorithm from Section 2 to work for discontiguous phrases.

Let us make some notations to ease the exposition of our phrase extraction algorithm. Let a, b and c be words in the source language, X a non-terminal used to denote the gaps in translation rules and α and β source phrases containing zero or more occurrences of X. Let M_α be the set of matches of the phrase α in the source side of the training corpus, where a phrase match is defined by a sequence of indices marking the positions where the contiguous subphrases of α are found in the training data. Our goal is to find M_α for every phrase α. Section 2 shows how to achieve this if X does not occur in α.

Let us consider the case when α contains at least one non-terminal. If $\alpha = X\beta$ or $\alpha = \beta X$, then $M_\alpha = M_\beta$, because the phrase matches are defined only in terms of the indices where the contiguous subpatterns match the training data. The words spanned by the leading or trailing non-terminal are not relevant because they do not appear in the translation rule. Since $|\beta| < |\alpha|$, M_β is already available in the cache as a consequence of the breadth first search approach we use to compute the sets M.

The remaining case is $\alpha = a\beta c$, where both $M_{a\beta}$ and $M_{\beta c}$ have been computed at a previous step. We take into consideration two cases depending on whether the next-to-last symbol of α is a terminal or not (i.e. $\alpha = a\beta bc$ or $\alpha = a\beta Xc$, respectively). In the former case, we calculate M_α by iterating over all the phrase matches in $M_{a\beta b}$ and selecting those matches that are followed by the word c. In the sec-

ond case, we take note of the experimental results of Lopez (2008a) who shows that translation rules that span more than 15 words have no effect on the overall quality of translation. In our implementation, we introduce a parameter `max_rule_span` for setting the maximum span of a translation rule. For each phrase match in $M_{\alpha\beta X}$, we check if any of the following `max_rule_span` words is c (subject to sentence boundaries and taking into account the current span of $\alpha\beta X$) and insert any new phrase matches in M_α accordingly. Note that M_α can also be computed by considering two cases based on the second symbol in α (i.e. $\alpha = ab\beta c$ or $\alpha = aX\beta c$) and by searching the word a at the beginning of the phrase matches in $M_{b\beta c}$ or $M_{X\beta c}$. In our implementation, we consider both options and apply the one that is likely to lead to a smaller number of comparisons. The complexity of the algorithm for computing $M_{\alpha=a\beta c}$ is $O(\min(|M_{a\beta}|, |M_{\beta c}|))$.

Lopez (2007) presents a similar grammar extraction algorithm for discontiguous phrases, but the complexity for computing M_α is $O(|M_{a\beta}| + |M_{\beta c}|)$. Lopez (2007) introduces a separate optimization based on double binary search (Baeza-Yates, 2004) of time complexity $O(\min(|M_{a\beta}|, |M_{\beta c}|) \log \max(|M_{a\beta}|, |M_{\beta c}|))$, designed to speed up the extraction algorithm when one of the lists is much shorter than the other. Our approach is asymptotically faster than both algorithms. In addition to this, we do not require the lists M_α to be sorted, allowing for a much simpler implementation. (Lopez (2007) needs van Emde Boas trees and an inverted index to efficiently sort these lists.)

The extraction algorithm can be optimized by precomputing an index for the most frequent discontiguous phrases (Lopez, 2007). To construct the index, we first need to identify the set of the most frequent K contiguous phrases in the training data, where K is an argument for our extraction tool. We use the LCP array (Manber and Myers, 1990), an auxiliary data structure constructed in linear time from a suffix array (Kasai et al., 2001), to find all the contiguous phrases in the training data that occur above a certain frequency threshold. We add these phrases to a max-heap together with their frequencies and extract the most frequent K contiguous patterns. We iterate over the source side of the training data and populate the index with all the discontiguous phrases of the form uXv and $uXvXw$, where u, v and w are amongst the most frequent K contiguous phrases in the training data.

4. Usage and implementation details

Our grammar extractor is designed as a standalone tool which takes as input a word-aligned parallel corpus and a test set and produces as output the set of translation rules applicable to each sentence in the test set. The extractor produces the output in the format expected by the `cdec` decoder, but the implementation is self-contained and easily extendable to other hierarchical phrase-based translation systems.

Our tool performs grammar extraction in two steps. The preprocessing step takes as input the parallel corpus and the file containing the word alignments and writes to disk binary representations of the data structures needed in the extraction step: the

symbol table, the source suffix array, the target data array, the word alignment, the precomputed index of frequent discontiguous phrases and a translation table storing estimates for the conditional word probabilities $p(s|t)$ and $p(t|s)$, for every source word s and target word t collocated in the same sentence pair in the training data. The translation probabilities are required for the scoring features in the extraction step. The output of the preprocessing step is written to disk in a directory specified by the user. A configuration file is also produced to reduce the number of parameters the user has to provide to the extraction step. The preprocessed data structures can be reused when extracting grammars for different test sets. The extraction step takes as input the precomputed data structures and a test corpus and produces a set of grammar files containing the applicable translation rules for each sentence in the test set. Note that our extraction tool expects the entire test corpus as input only to match the intended overall usage of the `cdec` pipeline and that the tool itself at no point takes advantage of the fact that the whole test corpus is known in advance.

Our extractor is written in `C++`. Compiling the code yields two binaries, `sacompile` and `extract`, corresponding to the two steps described above. `sacompile` takes the following parameters:

- `--help`: Prints a list of available options.
- `--source`: The path to the file containing the source side of the parallel corpus, one sentence per line.
- `--target`: The path to the file containing the target side of the parallel corpus, one sentence per line.
- `--bitext`: The path to the parallel corpus, one pair of sentences per line. The expected format is `source_sentence ||| target_sentence`. This parameter needs to be set only if `--source` and `--target` are not provided.
- `--alignment`: The path to the word alignment file. The expected format is the same as the one used by tools like `cdec` or `Moses` [2].
- `--output`: The directory where the binary representations are written.
- `--config`: The path where the config file will be created.
- `--max_rule_span`: The maximum number of words spanned by a rule.
- `--max_symbols`: The maximum number of symbols (words and non-terminals symbols) in the source side of a rule.
- `--min_gap_size`: The minimum number of words spanned by a non-terminal.
- `--frequent`: The number of frequent contiguous phrases to be extracted for the construction of the precomputed index.
- `--super_frequent`: The number of super frequent contiguous phrases to be used in the construction of the precomputed index (a subset of the contiguous phrases extracted with the `--frequent` parameter). Discontiguous phrases of the form $uXvXw$ are added to the index only if either both u and v or v and w are super-frequent.

[2] More details here: `http://www.cdec-decoder.org/guide/fast_align.html`

- `--min_frequency`: The minimum number of times a phrase must occur in the corpus to be considered a candidate for the set of most frequent phrases.
- `--max_phrase_len`: The maximum number of words spanned by a frequent contiguous phrase.

The `extract` binary takes the following parameters:
- `--help`: Prints a list of available options.
- `--config`: The path to the configuration file produced by the preprocessing step.
- `--grammars`: The directory where the files containing the translation rules for each sentence are written.
- `--threads`: The number of threads used for parallel extraction.
- `--max_rule_span`: The maximum number of words spanned by a rule.
- `--max_rule_symbols`: The maximum number of symbols (words and non-terminal symbols) in the source side of a rule.
- `--min_gap_size`: The minimum number of words spanned by a non-terminal.
- `--max_nonterminals`: The maximum number of non-terminals in a rule.
- `--max_samples`: A threshold on the number of phrase matches used to extract translation rules for each phrase.
- `--tight_phrases`: Use tight constraints for extracting rules (Chiang, 2007).
- `--leave_one_out`: If the training set is used as a test set, the extractor will ignore any phrase matches in the test sentence for which the rules are extracted.

The `extract` binary reads the test corpus from standard input and produces an summary file at standard output. For both binaries, the only required parameters are the files and directories required for input and output, while the remaining parameters are initialized with sensible default values.

Our implementation leverages the benefits of a multithreaded environment to speed up grammar extraction. The test corpus is distributed dynamically across the number of available threads (specified by the user with the `--threads` parameter). All the data structures computed in the preprocessing step are immutable during extraction and can be effectively shared across multiple threads at no additional time or memory cost. In contrast, the existing extractor (implementing Lopez (2008a)'s algorithm) available in `cdec` uses a multi-process approach to parallel extraction. This is ill-suited for memory constrained environments because the preprocessed data structures are copied across all the processes used for extraction. As a result, the amount of memory available will restrict the degree of parallelization that the extractor can achieve.

Our code is released together with a suite of unit tests based on the `Google Test` and `Google Mock` frameworks. The unit tests are provided to encourage developers to add their own features to our grammar extractor without the fear that their changes might have unexpected consequences.

Implementation	Time (minutes)	Memory (GB)
Original `cython` extractor	28.518	6.4
`C++` reimplementation	2.967	6.4
Current work (`C++`)	2.903	6.3

Table 1. Results for the preprocessing step.

Implementation	Time (minutes)	Memory (GB)
Original `cython` extractor	309.725	4.4
`C++` reimplementation	381.591	6.4
Current work (`C++`)	75.496	5.7

Table 2. Results for the phrase extraction step.

5. Experiments

In this section, we present a set of experiments which illustrate the benefits of our new extractor. We compare our implementation with the one available in `cdec` which implements the algorithm proposed by Lopez (2008a). The existing extractor is written in `cython`. In order to make the comparison fair and to prove that the speed ups we obtain are indeed a result of our new algorithm, we also report results for an implementation of Lopez (2008a)'s algorithm in `C++`.

For our experiments, we used the French-English data from the `europarl-v7` corpus, a set of 2,002,756 pairs of sentences containing a total of 104,722,300 tokens. The training corpus was tokenized, lowercased and pairs of sentences with unusual length ratios were filtered out using the corpus preparation scripts available in `cdec`[3]. The corpus was aligned using `fast_align` (Dyer et al., 2013) and the alignments were symmetrized using the `grow-diag-final-and` heuristic. We extracted translation rules for the `newstest2012` test corpus[4]. The test corpus consists of 3,003 sentences and was tokenized and lowercased using the same scripts as the training corpus.

Table 1 shows results for the preprocessing step of the three implementations. We note a 10-fold time reduction when reimplementing Lopez (2008a)'s algorithm in `C++`. We believe this is a consequence of inefficient programming when the precomputed index is constructed in the `cython` code and not a result of using different programming languages. Our new implementation does not significantly outperform an efficient implementation of the preprocessing step of Lopez (2008a)'s extractor because it computes the same set of data structures.

[3]We followed the indications provided here: `http://www.cdec-decoder.org/guide/tutorial.html`.

[4]The test corpus is available here: `http://www.statmt.org/wmt14/translation-task.html`.

Implementation	Time (minutes)	Memory (GB)
Original `cython` extractor	37.950	35.2
`C++` reimplementation	51.700	10.1
Current work (`C++`)	9.627	6.1

Table 3. Results for parallel extraction using 8 processes/threads.

The second set of results (Table 2) show the running times and memory requirements of the extraction step. Our `C++` reimplementation of Lopez (2008a)'s algorithm is slightly less efficient than the original `cython` extractor, supporting the idea that the two programming languages have roughly similar performance. We note that our novel extraction algorithm is over 4 times faster than the original approach of Lopez (2008a). The increased memory usage is not a real concern because it does not exceed the amount of memory used in the preprocessing step.

Table 3 demonstrates the benefits of parallel phrase extraction. We repeated the experiments from Table 2 using 8 processes in `cython` and 8 threads in `C++`. As expected, the running times decrease roughly 8 times. The benefits of shared-memory parallelism are evident, our new implementation is saving 29.1 GB of memory. Our implementation continues to use less memory than the preprocessing step even when running in multithreaded mode.

In conclusion, this paper presents an open source implementation of a SCFG extractor integrated with `cdec` that is 4 times faster than the existing extractor (Lopez, 2008a) and that is better designed for parallel environments. Compared to traditional phrase tables, our approach is considerably more memory efficient without involving any pruning based on the test corpus, therefore scaling to unseen sentences.

Bibliography

Baeza-Yates, Ricardo A. A fast set intersection algorithm for sorted sequences. In *Combinatorial Pattern Matching*, pages 400–408. Springer Berlin Heidelberg, 2004.

Callison-Burch, Chris, Colin Bannard, and Josh Schroeder. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 255–262, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

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

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '13)*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Gusfield, Dan. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, USA, 1997.

Kasai, Toru, Gunho Lee, Hiroki Arimura, Setsuo Arikawa, and Kunsoo Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Combinatorial Pattern Matching*, pages 181–192. Springer Berlin Heidelberg, 2001.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '03)*, pages 48–54. Association for Computational Linguistics, 2003.

Larsson, N. Jesper and Kunihiko Sadakane. Faster suffix sorting. *Theoretical Computer Science*, 387(3):258–272, 2007.

Lopez, Adam. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '07)*, pages 976–985, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Lopez, Adam. *Machine translation by pattern matching*. ProQuest, 2008a.

Lopez, Adam. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49, 2008b.

Manber, Udi and Gene Myers. Suffix arrays: A new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 319–327. Society for Industrial and Applied Mathematics, 1990.

Och, Franz Josef and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

Schwartz, Lane and Chris Callison-Burch. Hierarchical phrase-based grammar extraction in joshua. *The Prague Bulletin of Mathematical Linguistics*, 93(1), 2010.

Zens, Richard and Hermann Ney. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '07)*, pages 492–499, Rochester, New York, 2007. Association for Computational Linguistics.

Zhang, Ying and Stephan Vogel. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT-05)*, pages 30–31, 2005.

Address for correspondence:
Paul Baltescu
paul.baltescu@cs.ox.ac.uk
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 27–36

Tree Transduction Tools for cdec

Austin Matthews[a], Paul Baltescu[b], Phil Blunsom[b], Alon Lavie[a], Chris Dyer[a]

[a] Carnegie Mellon University
[b] University of Oxford

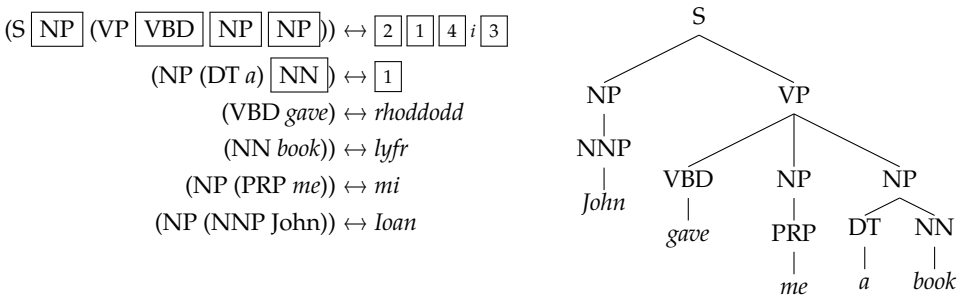
Abstract

We describe a collection of open source tools for learning tree-to-string and tree-to-tree transducers and the extensions to the cdec decoder that enable translation with these. Our modular, easy-to-extend tools extract rules from trees or forests aligned to strings and trees subject to different structural constraints. A fast, multithreaded implementation of the Cohn and Blunsom (2009) model for extracting compact tree-to-string rules is also included. The implementation of the tree composition algorithm used by cdec is described, and translation quality and decoding time results are presented. Our experimental results add to the body of evidence suggesting that tree transducers are a compelling option for translation, particularly when decoding speed and translation model size are important.

1. Tree to String Transducers

Tree-to-string transducers that define relations on strings and trees are a popular formalism for capturing translational equivalence where syntactic tree structures are available in either the source or target language (Graehl et al., 2008; Galley et al., 2004; Rounds, 1970; Thatcher, 1970). The tools described in this paper are a restricted version of top-down tree transducers that support multi-level tree fragments on one side and strings on the other, with no copying or deletion (Huang et al., 2006; Cohn and Blunsom, 2009). Such transducers can elegantly capture syntactic regularities in translation. For example see Fig. 1, which gives the rules necessary to translate between English (an SVO language with ditransitive verbs) and Welsh (a VSO language with prepositional datives). In our notation, transducers consist of a set of rules

 Corresponding author: cdyer@cs.cmu.edu
Cite as: Austin Matthews, Paul Baltescu, Phil Blunsom, Alon Lavie, Chris Dyer. Tree Transduction Tools for cdec. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 27–36.
doi: 10.2478/pralin-2014-0011.



cdec text format of above transducer (with example features):

```
(S [NP] (VP [VBD] [NP] [NP])) ||| [2] [1] [4] i [3] ||| logP(s|t)=-0.2471
(NP (DT a) [NN]) ||| [1] ||| logP(s|t)=-0.6973 Delete_a=1
(VBD gave) ||| rhoddodd ||| logP(s|t)=-2.3613
(NN book) ||| lyfr ||| logP(s|t)=-0.971
(NP (PRP me)) ||| mi ||| logP(s|t)=-1.3688
(NP (NNP John)) ||| Ioan ||| logP(s|t)=0
```

cdec input text format of above tree:

```
(S (NP (NNP John)) (VP (VBD gave) (NP (PRP me)) (NP (DT a) (NN book))))
```

Figure 1. Example single-state transducer that transduces between the SVIO English tree (upper right of figure) and its VSOP Welsh translation: rhoddodd Ioan lyfr i mi.

(also called edges) which pair a tree fragment in one language with a string of terminal symbols and variables in a second language. Frontier nonterminal nodes in the tree fragment are indicated with a box around the nonterminal symbol, and the corresponding substitution site in the string is indicated by a box around a number indexing the nonterminal variable in the tree fragment (counting in top-down, left to right, depth first order). Additionally, tree-to-string transducers can be further generalized so as to have multiple transducer states, shown in Fig. 2. The transducers in Fig. 1 can be understood to have a single state. For formal properties of tree-to-string transducers, we refer the reader to the above citations.

Tree-to-string transducers define a relation on strings and trees and, in translation applications, are capable of transforming either source trees (generated by a parser) into target language strings or source strings into target-language parse trees. Running the transducer in the tree-to-string direction can avail itself of specialized algorithms similar to finite state composition (§4); in the string-to-tree direction, they can be trivially converted to synchronous context free grammars and transduction can be carried out with standard CFG parsing algorithms (Galley et al., 2004).

$$q_0 : (\text{S } \boxed{\text{NP}} \text{ (VP } \boxed{\text{VB}} \boxed{\text{NP}})) \leftrightarrow \boxed{1} : q_0 \boxed{2} : q_0 \boxed{3} : q_{acc}$$
$$q_0 : (\text{NP (DT } \textit{the}) \boxed{\text{NN}}) \leftrightarrow \textit{der} \boxed{1} : q_0$$
$$q_{acc} : (\text{NP (DT } \textit{the}) \boxed{\text{NN}}) \leftrightarrow \textit{den} \boxed{1} : q_0$$
$$q_0 : (\text{NN } \textit{dog}) \leftrightarrow \textit{Hund}$$

cdec text format of above transducer (with example features):

```
[Q0] ||| (S [NP] (VP [VB] [NP])) ||| [Q0,1] [Q0,2] [QACC,3] ||| lp=-2.9713
[Q0] ||| (NP (DT the) [NN]) ||| der [Q0,1] ||| lp=-1.3443
[QACC] ||| (NP (DT the) [NN]) ||| den [Q0,1] ||| lp=-2.9402
[Q0] ||| (NN dog) ||| Hund ||| lp=-0.3171
```

Figure 2. A tree-to-string transducer with multiple states encoding structural information for choosing the proper nominal inflection in English–German translation.

2. Heuristic Hypergraph-based Grammar Extraction

In this section we describe a general purpose tree-to-string and tree-to-tree rule learner. We will consider the tree-to-tree alignment problem in this case (the tree-to-string case is a straightforward simplification). Instead of extracting rules from a pair of aligned trees, rules from a pair of aligned *hypergraphs* (any tree can easily be transformed into an equivalent hypergraph; an example of such a hypergraph is shown in Fig. 3). By using hypergraphs, the rule extraction algorithm can use forest outputs from parsers to capture parse uncertainty; furthermore (as discussed below), it simplifies the rule extraction algorithm so that extraction events—even of so-called "composed rules" (Galley et al., 2006)—always apply locally to a single edge rather than considering larger structures. This yields a simpler implementation.

The rule extraction process finds pairs of aligned nodes in the hypergraphs based on the terminal symbol alignment. We will call a source node S and a target node T *node-aligned* if the following conditions hold. First, S and T must either both be non-terminals or both be terminals. Aligning a terminal to a non-terminal or vice-versa is disallowed. Second, there must be at least one alignment link from a terminal dominated by S to a terminal dominated by T. Third, there must be no alignment links from terminals dominated by S to terminals outside of T or vice-versa.

We define a "rule" to be pair of hyperedges whose heads are node-aligned *and* whose non-terminal children are node-aligned in a one-to-one (bijective) manner. For example, in the sample tree, we see that the source node $\texttt{PP}_{4,6}$ is node-aligned to the target node $\texttt{PP}_{5,7}$ *and* their children are node-aligned $\texttt{TO}_{4,5}$ to $\texttt{PREP}_{5,6}$ and $\texttt{NN}_{5,6}$ to $\texttt{NP}_{5,7}$. This edge pair corresponds to the rule `[PP::PP]` $\rightarrow$ `[TO,1] [NN,2] ||| [PREP,1] [NP,2]`. Note in this formalism, no edges headed by terminals, so we will not extract any rules with terminal "heads".

The above formulation allows the extraction of so-called "minimal" rules that do not contain internal structure, but it does not yet include any mechanism for extracting more complex rules. Rather than adding extra mechanisms to the rule extractor, we create extra edges in the hypergraph so that "composed" edges are available to the extractor. To do so, we recursively add edges from non-overlapping sets of descendent nodes. For example, one hyperedge added to the source side of the sample hypergraph pair is $\texttt{VP}_{3,6} \rightarrow \texttt{walked TO}_{4,5}\ \text{NN}_{5,6}$. Independently, on the target side we add an edge $\texttt{VP}_{3,7} \rightarrow \texttt{a marché PREP}_{5,6}\ \text{NP}_{5,7}$.

Now when we extract rules, we will find these two edges will give rise to the rule `[VP::VP]` → `walked [TO,1] [NN,2] ||| a marché [PREP,1] [NP,2]`, a composed rule not extractable by the bald algorithm.

While using composed edges allows us to extract all permissible rules from a pair of aligned trees, to be consistent with previous work, we introduce one more type of hypergraph augmentation. Hanneman et al. (2011) allow for adjacent sibling non-terminal nodes to be merged into one *virtual node*, which may then be node-aligned to opposite nodes, be they "real" or virtual. To enable this, we explicitly add virtual nodes to our hypergraph and connect them to their children with a hyperedge. Furthermore, for every hyperedge that contained all of the sibling nodes as non-terminal tails, we add a duplicate hyperedge that uses the new virtual node instead.

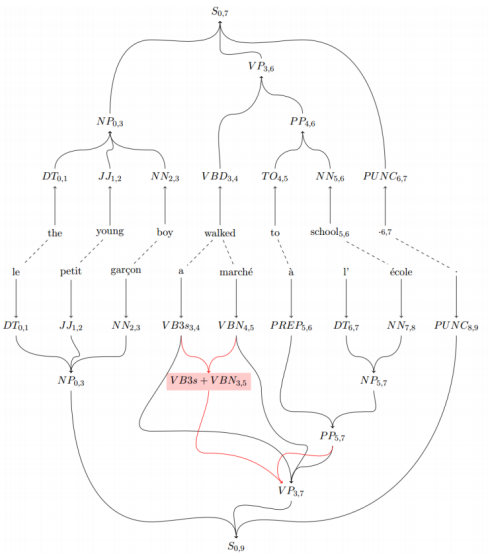


Figure 3. A pair of aligned hypergraphs. $NP_{0,3}$ represents an NP over the span $[0, 3)$. An example virtual node and its corresponding virtual edges is shown in red.

For example, in Fig. 3, we have added a new non-terminal node labeled $\texttt{VB3s+VBN}_{3,5}$ to the hypergraph. This node represents the fusion of the $\texttt{VB3s}_{3,4}$ and $\texttt{VBN}_{4,5}$ nodes. We then add a hyperedge headed by the new $\texttt{VB3s+VBN}_{3,5}$ with tails to both $\texttt{VB3s}_{3,4}$ and $\texttt{VBN}_{4,5}$. Furthermore, we make a copy of the edge $\texttt{VP}_{3,7} \rightarrow \texttt{VB3s}_{3,4}\ \texttt{VBN}_{4,5}\ \texttt{PP}_{5,7}$, and replace the $\texttt{VB3s}_{3,4}$ and $\texttt{VBN}_{4,5}$ tail nodes with a single tail, $\texttt{VB3s+VBN}_{3,5}$, to form the new edge $\texttt{VP}_{3,7} \rightarrow \texttt{VB3s+VBN}_{3,5}\ \texttt{PP}_{5,7}$. The addition of this new hyperedge allows the extraction of the rules `[VBD::VB3s+VBN]` → `walked ||| a marché` and `[VP::VP]` → `[VBD,1] [PP,2]`

`|||` `[VB3s+VBN,1]` `[PP,2]`, both of which were unextractable without the virtual node.

With the addition of virtual nodes, our work is directly comparable to Hanneman et al. (2011), while being more modular, extensible and provably correct. One particularly interesting extension our hypergraph formulation allows is the use of weighted parse *forests* rather than 1-best trees. This helps our rule extractor to overcome parser errors and allows us to easily handle cases of ambiguity, in which two or more trees may be equally likely for a given input sentence.

3. Bayesian Synchronous Tree to String Grammar Induction

Although HyperGrex, the tool described in the previous section, is flexible, it relies on heuristic word alignments that were generated without knowledge of the syntactic structure or the final translation formalism they will be used in. In this section, we present our open source implementation of the synchronous tree-to-string grammar induction algorithm proposed by Cohn and Blunsom (2009).[1] This model directly reasons about the most likely tree-to-string grammar that explains the parallel corpus. Tree-to-tree grammars are not currently supported.

The algorithm relies on a Bayesian model which incorporates a prior preference for learning small, generalizable STSG rules. The model is designed to jointly learn translation rules and word alignments. This is important for capturing long distance reordering phenomena, which might otherwise be poorly modeled if the rules are inferred using distance penalized alignments (e.g. as in the heuristic proposed by Galley et al. (2004) or the similar one used by HyperGrex).

The model represents the tree-to-string grammar as a set of distributions $\{G_c\}$ over the productions of each non-terminal c. Each distribution G_c is assumed to be generated by a Dirichlet Process with a concentration parameter α_c and a base distribution $P_0(\cdot \mid c)$, i.e. $G_c \sim DP(\alpha_c, P_0(\cdot \mid c))$. The concentration parameter α_c controls the model's tendency towards reusing rules or creating new ones according to the base distribution and has a direct influence on the size of the resulting grammar. The base distribution is defined to assign probabilities to an infinite set of rules. The probabilities decrease exponentially as the sizes of the rules increase, biasing the model towards learning smaller rules.

Instead of representing the distributions G_c explicitly, we integrate over all the possible values of G_c. We obtain the following formula for estimating the probability of a rule r with root c, given a fixed set of derivations $\mathbf{r}$ for the training corpus:

$$p(r \mid \mathbf{r}, c; \alpha_c, P_0) = \frac{n_r + \alpha_c P_0(r \mid c)}{n_c + \alpha_c}, \tag{1}$$

where n_r is the number of times r occurs in $\mathbf{r}$ and n_c is the number of rules with root c in $\mathbf{r}$.

[1] Our code is publicly available here: `https://github.com/pauldb89/worm`.

Cohn and Blunsom (2009) train their model using Gibbs sampling. To simplify the implementation, an alignment variable is defined for every internal node in the parsed corpus. An alignment variable specifies the interval of target words which are spanned by a source node. Alternatively, a node may not be aligned to any target words or may span a discontiguous group of words, in which case it is annotated with an empty interval. Non-empty alignment variables mark the substitution sites for the rules in a derivation of a parse tree. Overall, they are used to specify a set of sampled derivations $\mathbf{r}$ for the entire training data. Alignment spans are constrained to subsume the spans of their descendants and must be contained within the spans of their ancestors. In addition to this, sibling spans belonging to the frontier of the same rule must not overlap.

We implement Gibbs sampling with the help of two operators: `expand` and `swap`. The `expand` operator works by resampling a randomly selected alignment variable a, while keeping all the other alignment variables fixed. The set of possible outcomes consists of the empty interval and all the intervals assignable to a such that the previous conditions continue to hold. Each outcome is scored proportionally to the new rules it creates, using Equation 1, conditioned on all the rules in the training data that remain unaffected by the sampling operation. The `swap` operator randomly selects two frontier nodes labelled with non-terminals belonging to the same STSG rule and chooses to either swap their alignment variables or to leave them unchanged. The outcomes are weighted similarly to the previous case. The goal of the `swap` operator is to improve the sampler's ability to mix, especially in the context of improving word reordering, by providing a way to execute several low probability `expand` steps at once.

Our implementation of the grammar induction algorithm is written in `C++`. Compiling the code results in several binaries, including `sampler`, which implements our Gibbs sampler. Our tool takes as input a file containing the parse trees for the source side of the parallel corpus, the target side of the parallel corpus, the word alignments of the training data, and two translation tables giving $p(s \mid t)$ and $p(t \mid s)$ respectively. The word alignments are needed only to initialize the sampler with the first set of derivations (Galley et al., 2004). The remaining input arguments (hyperparameters, rule restrictions, etc.) are initialized with sensible default values. Running the binary with the `--help` option will produce the complete list of arguments and a brief explanation for each. The tool produces several files as output, including one containing the set of rules together with their probabilities, computed based on the last set of sampled derivations. The documentation released with our code shows how to prepare the training data, run the tool and convert the output to the `cdec` format.

Our tool leverages the benefits of a multithreaded environment to speed up grammar induction. At every sampler iteration, each training sentence is dynamically allocated to one of the available threads. In our implementation, we use a hash-based implementation of a Chinese Restaurant Process (CRP) (Teh, 2010) to efficiently compute the rule probabilities given by Equation 1. The data structure is updated when-

ever one of the `expand` or `swap` operators is applied. To lock this data structure with every update would completely cancel the effect of parallelization, as all the basic operations performed by the sampler are dependent on the CRP. Instead, we distribute a copy of the CRP on every thread and synchronize the data structures at the end of each iteration. Although the CRPs diverge during an iteration through the training data, no negative effects are observed when inferring STSGs in multithreaded mode.

4. Tree-to-string translation with `cdec`

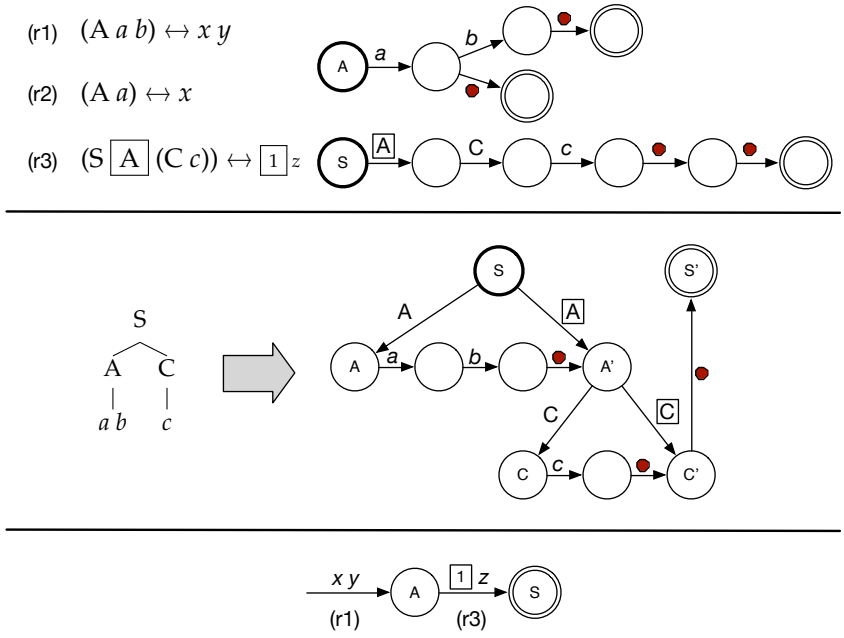


Figure 4. DFA representation of a tree transducer (above) and an input tree (middle). This transducer will transduce the input tree to the hypergraph (below) yielding a single string x y z*, using rules (r1) and (r3). Red octagons are closing parentheses.*

The `cdec` decoder (Dyer et al., 2010) has a modular decoder architecture that factors the decoding problem into multiple stages: first, a hypergraph is generated that represents the translation search space produced by composing the input (a string, lattice, or tree) with the relevant transducer (a synchronous context-free grammar, a finite state transducer, etc.); second, the hypergraph is rescored—and possibly restructured (in the case of adding an n-gram language model)—with generic feature

extractors; finally, various outputs of interest are extracted (the best translation, the k-best translations, alignments to a target string, statistics required for parameter optimization, etc.).

The original `cdec` implementation contained hypergraph generators based on a variety of translation backends, although SCFG translation is the most widely used (Chiang, 2007). In this section, we describe the tree-to-string algorithm that generates a translation forest given a source language parse tree and a tree-to-string transducer.

The construction of the hypergraph takes place by composing the input tree with the tree-to-string transducer using a top down, recursive algorithm. Intuitively, the algorithm matches all rules in the transducer that start at a given node in the input tree and match at least one complete rewrite in the source tree. Any variables that were used in the match are then recursively processed until the entire input tree is completed. To make this process efficient, the tree side of the input transducer is determinized by depth-first, left-to-right factoring—this process is analogous to left factoring a context-free grammar (Klein and Manning, 2001). By representing the tree using the same depth-first, left-to-right representation, standard DFA intersection algorithms can be used to compute each step of the recursion. The DFA representation of a transducer (tree side) and an input tree (starting at nonterminal S) is shown in Fig. 4.

To understand how this algorithm proceeds on this particular input, the input tree DFA is matched against the 'S' DFA. The output transductions are stored in the final states of the transducer DFA, and for all final states in the transducer DFA that are reached in a final state of the input DFA, an edge is added to the output hypergraph, one per translation option. Variables that were used in the input DFA are then recursively processed, starting from the relevant transducer DFA (in this case since first (r3) will be used which has an A variable, then 'A' DFA will then be invoked recursively).

Extractor	k_s	k_t	Instances	Types
`grex`	1	1	24.9M	11.8M
`HyperGrex`	1	1	25.9M	12.7M
`HyperGrex`	1	10	33.3M	17.7M
`HyperGrex`	10	1	33.7M	17.9M
`HyperGrex`	10	10	48.7M	24.3M

Figure 5. Grammar sizes using different grammar extraction set ups. k_s *(*k_t*) represents the number of source (target) trees used.*

5. Experiments

We tested our tree-to-tree rule learner on the FBIS Chinese–English corpus (LDC2003E14), which consists of 302,966 sentence pairs or 9,350,506 words on the English side. We first obtain k-best parses for both sides of FBIS using the Berkeley Parser[2] and align the corpus using `fastalign` (Dyer et al., 2013). We use a 5-gram language model built

[2]https://code.google.com/p/berkeleyparser/

with KenLM on version four of the English GigaWord corpus plus the target side of FBIS, smoothed with improved Kneser-Ney smoothing. For each set up we extract rules using `grex` (Hanneman et al., 2011) or our new tool. When using our tool we have the option of simply using 1-best trees to compare directly to `grex`, or using the weighted forests consisting of all of Berkeley's k-best parses on the source side, the target side, or both. For these experiments we use $k = 10$. Each system is tuned on mt06 using Hypergraph MERT. We then test each system on both mt03 and mt06.

Details concerning the size of the extracted grammars can be found in Table 5.[3] Translation quality results are shown in Table 6.

5.1. Baysiean Grammar Experiments

Extractor	k_s	k_t	mt06	mt03	mt08
`grex`	1	1	29.6	31.8	23.4
`HyperGrex`	1	1	30.1	32.4	24.0
`HyperGrex`	1	10	**30.4**	**32.9**	**24.3**
`HyperGrex`	10	1	29.5	32.0	23.1
`HyperGrex`	10	10	30.0	32.7	23.7

Figure 6. BLEU *results on mt06 (tuning set), mt03, and mt08 using various grammar extraction configurations.*

The Bayesian grammar extractor we describe is constructed to find compact grammars that explain a parallel corpus. We briefly discuss the performance of these grammars in a tree-to-string translation task relative to a standard Hiero baseline. Each of these systems was tuned on mt03 and tested on mt08. Table. 7 summarizes the findings. Although tree-to-string system with minimal rules underperforms Hiero slightly, it uses orders of magnitude fewer rules—in fact the number of rules in the Hiero grammar *filtered* for the 691-sentence test set is twice as large as the Bayesian grammar. The unfiltered Hiero grammar is 2 orders of magnitude larger than the Bayesian grammar.

Extractor	iterations	rule count	mt08
Hiero	–	36.6M	27.9
HyperGrex minimal (§2)	–	1.4M	26.5
Bayes (§3)	100	0.77M	26.5
Bayes (§3)	1,000	0.74M	26.9

Figure 7. Comparing HyperGrex (minimal rules), the Bayesian extractor after different numbers of iterations, and Hiero.

[3] This indicates that `grex` failed to extract certain valid rules. This conclusion was validated by our team, and confirmed with the authors of (Hanneman et al., 2011).

Acknowledgements

This work was supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, the Qatar National Research Fund (a member of the Qatar Foundation) under grant NPRP 09-1140-1-177, by a Google Faculty Research Grant (2012_R2_10), and by the NSF-sponsored XSEDE program under grant TG-CCR110017.

Bibliography

Chiang, David. Hierarchical phrase-based translation. *Computational Linguistics*, 2007.

Cohn, Trevor and Phil Blunsom. A bayesian model of syntax-directed tree to string grammar induction. In *Proc. of EMNLP*, 2009.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*, 2010.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. of NAACL*, 2013.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *HLT-NAACL*, 2004.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proc. of NAACL*, 2006.

Graehl, Jonathan, Kevin Knight, and Jonathan May. Training tree transducers. *Computational Linguistics*, 34(3), 2008.

Hanneman, Greg, Michelle Burroughs, and Alon Lavie. A general-purpose rule extractor for SCFG-based machine translation. In *Proc. of SSST*, 2011.

Huang, Liang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA*, 2006.

Klein, Dan and Christopher D. Manning. Parsing and hypergraphs. In *Proc. of IWPT*, 2001.

Rounds, William C. Mappings and grammars on trees. *Mathematical Systems Theory*, 4(3):257–287, 1970.

Teh, Yee Whye. Dirichlet process. In *Encyclopedia of Machine Learning*, pages 280–287. 2010.

Thatcher, James W. Generalized sequential machine maps. *Journal of Computer and System Sciences*, 4:339–367, 1970.

Address for correspondence:
Chris Dyer
cdyer@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, United States

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 37–46

The Machine Translation Leaderboard

Matt Post, Adam Lopez

Human Language Technology Center of Excellence, Johns Hopkins University

Abstract

Much of an instructor's time is spent on the management and grading of homework. We present the Machine Translation Leaderboard, a platform for managing, displaying, and automatically grading homework assignments. It runs on Google App Engine, which provides hosting and user management services. Among its many features are the ability to easily define new assignments, manage submission histories, maintain a development / test set distinction, and display a leaderboard. An entirely new class can be set up in minutes with minimal configuration. It comes pre-packaged with five assignments used in a graduate course on machine translation.

1. Introduction

Much of an instructor's time is spent on the management and grading of homework. For many types of learning, such as the grading of essays, this time is a necessary and critical component of the learning process. But there are also many types of assignments that are easily automatable. Time spent grading them, and in managing assignment infrastructure (whether physical or digital), are drains on the instructor's limited resources which could be better spent elsewhere.

For homework assignments in the data sciences, grading can be automated using sites like `kaggle.com`, where students can upload solutions to empirical problems posed by instructors. Unfortunately, kaggle does not allow instructors to use custom evaluation measures, which makes it a poor fit for fields like machine translation that use idiosyncratic, domain-specific metrics.

To help with this, we present the Machine Translation Leaderboard (MTL), a platform for managing, displaying, and automatically grading homework assignments. It runs on Google App Engine, which provides hosting, authentication, and user

 Corresponding author: post@cs.jhu.edu
Cite as: Matt Post, Adam Lopez. The Machine Translation Leaderboard. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 37–46. doi: 10.2478/pralin-2014-0012.

management services tied to a Google account. Students use a web interface to upload the results of their assignments (not code), which are then automatically graded and displayed. Among the MTL's many features are the ability to easily define new assignments, manage student submissions and submission histories, automatically grade them, maintain a development / test set distinction, and display a competitive leaderboard. An entirely new class can be setup in minutes, with minimal configuration. Packaged with the MTL are five assignments: alignment, decoding, evaluation, reranking, and inflection. Each of these assignments includes baseline (default) outputs and upper bounds, and provides ample room for improvement on their metrics through student exploration of both standard and novel approaches to these classic machine translation problems.

The MTL served as the foundation of a combined graduate and undergraduate course in machine translation, taught by the authors at Johns Hopkins University in the spring of 2014. The time we saved allowed us to focus on other aspects of teaching, including a course emphasis on scientific writing.

2. Quick-Start Guide

A new class with the default set of assignments can be setup in minutes, with minimal configuration.

1. Create an account at `appengine.google.com`
2. Create a new application. The name ("Application Identifier") you choose will become part of the URL for your class, e.g., "leaderboard" will result in a class URL of `leaderboard.appspot.com`. We will use the variable `$APPID` to refer to your choice.
3. Install the Google App Engine SDK for Python from `https://developers.google.com/appengine/downloads`
4. Clone the repository

```
$ git clone https://github.com/mjpost/leaderboard.git
$ cd leaderboard
```

5. Edit `app.yaml`, changing the value of the "application" key to the identifier you chose above

```
application: $APPID
```

6. Deploy.

```
$ appcfg.py --oauth2 update .
```

The first time you do this, a web browser window will open, prompting you to authenticate.

You can now access the leaderboard interface for uploading assignment results by loading your course's base URL (`$APPID.appspot.com`), and you can access the leaderboard itself at `$APPID.appspot.com/leaderboard.html`.

2.1. Administrative accounts

By default, the Google account used to create and host the leaderboard is the administrative account. This account has the following special permissions:

- Viewing submissions from all students, regardless of their privacy settings.
- Viewing scores for all submissions on hidden test data.
- Submitting baseline, default, and oracle entries.
- Accessing back-end administration.

You may wish to grant administrative permission to co-instructors and TAs:

1. Navigate to the back-end administration page:
   `appengine.google.com/dashboard?&app_id=s~$APPID`
2. From the navigation menu, click on "Permissions" under "Administration"
3. Add the Google accounts of your colleagues

2.2. Setting Deadlines

By default, only the first assignment is enabled, and its deadline has already passed. To enable assignments, you must do three things:

1. Uncomment the assignment's scorer in the file `leaderboard.py`.

```
scorer = [
  scoring.upload_number,
  # scoring.alignment,
  # scoring.decode,
  # scoring.evaluation,
  # scoring.rerank,
  # scoring.inflect,
]
```

2. Adjust the assignment's due date. Edit `scoring/upload_number.py` and set the value of the `deadline` variable:

```
deadline = datetime.datetime(2014, 07, 17, 23, 59)
```

3. Deploy the changes:

```
$ appcfg.py --oauth2 update .
```

2.3. The Leaderboard

Visit `$APPID.appspot.com/leaderboard.html` to view the leaderboard. It displays a grid whose columns are assignments and whose rows are student entries. The rows are sorted from best to worst according to the most recent assignment.

Note that students have the option to hide their results from the leaderboard. This settings only hides their results from other students; accessed from an administrative account, the leaderboard displays everyone's results, denoting hidden students with strikeout text. Therefore, if you are displaying the leaderboard on a project, be sure to logout from your `appspot.com` account before accessing the leaderboard.

3. Modifying and Creating Assignments

3.1. Data Model and API

We have pre-packaged five assignments with the leaderboard, but it is easy to add new assignments. Before doing so, it is useful to understand the basic data model and API implemented in `leaderboard.py`. It defines two types of database records using Google's NDB (entity database) API. The first is a Handle record, which corresponds to a user that appears on the leaderboard.

```
class Handle(ndb.Model):
  user = ndb.UserProperty() # handle with no user belongs to admins
  leaderboard = ndb.BooleanProperty()
  handle = ndb.TextProperty()
  submitted_assignments = ndb.BooleanProperty(repeated=True)
```

Handles are managed by the leaderboard code and need not be modified by assignment code. New assignment types are more likely to interact with the Assignment record, which corresponds to a single student submission for an assignment.

```
class Assignment(ndb.Model):
  handle = ndb.KeyProperty()
  number = ndb.IntegerProperty()
  filename = ndb.StringProperty()
  filedata = ndb.BlobProperty()
  score = ndb.FloatProperty()
  test_score = ndb.FloatProperty()
```

```
percent_complete = ndb.IntegerProperty()
timestamp = ndb.DateTimeProperty(auto_now_add=True)
```

When a student uploads a solution to an assignment, the leaderboard code sets several of these fields, including `handle`, `number`, `filename`, `filedata`, and `timestamp`. An assignment will mostly interact with `score`, `test_score`, and `percent_complete`. Rather than modify these fields directly, it is preferred to modify them through a callback function that each new assignment type must provide with the following signature:

```
def score(data, assignment_key, test=False)
```

When a student uploads their results, this function is invoked twice: once with `test=False` and once with `test=True`, to provide development and test set scores, respectively. Each call provides the full contents of the uploaded file in the `data` field and the NDB key of the new assignment record in the `assignment_key` field. Although the callback may use this key to update the `score` or `test_score` fields directly, this is not the preferred way to update scores. Instead, `score(...)` should return it as the first value of a two-element tuple that the caller expects. The second element of the tuple is used to update `percent_complete` of the Assignment record. So, if an assignment can be scored quickly, the correct behavior is to simply return `(s, 100)`, where `s` is the computed score.

However, in special cases computing an assignment's score may require some time to compute, and the leaderboard provides functionality to handle this. In this case, the `score(...)` callback may return `(float("-inf"), 0)`, to indicate that the score has not yet been computed. The callback should then invoke a new background task to complete the scoring behavior, and this function should update `percent_complete` periodically until the score is computed. The leaderboard uses this information to display a progress bar to the student. An example can be found in `scoring/decode.py`.

3.2. Pre-packaged Assignments

The Leaderboard comes pre-packaged with five assignments. Instructions for each assignment appear on our course webpages (`http://mt-class.org/jhu/`), under the Homework tab.

3.2.1. Assignment 1: Align

In this assignment, the input is a parallel text and students must produce word-to-word alignments.[1] The pre-packaged version of the assignment scores alignments

[1] `http://mt-class.org/jhu/hw1.html`

Table 1. List of included assignments. All file locations are relative to the `scoring` subdirectory.

Assignment	Description	File
Setup	Make sure everything is working	`upload_number.py`
Alignment	Implement a word aligner	`alignment.py`
Decoding	Maximize the model score of a decoder	`decode.py`
Evaluation	Choose the better of MT system outputs	`evaluation.py`
Reranking	Rerank k-best lists by adjusting feature weights	`rerank.py`
Inflection	Choose the appropriate inflection for each of a sequence of lemmas	`inflect.py`

against 484 manually aligned sentences of the Canadian Hansards.[2] The alignments were developed by Och and Ney (2000), which we obtained from the shared task resources organized by Mihalcea and Pedersen (2003). We use the first 37 sentences of the corpus as development data and the remaining 447 as test. The scorer is implemented in `scoring/alignment.py` with data in `scoring/alignment_data`. To score against a different dataset, simply change the data files.

3.2.2. Assignment 2: Decode

In this assignment, the input is a fixed translation model and a set of input sentences, students and they must produce translations with high model score.[3] The model we provide is a simple phrase-based translation model (Koehn et al., 2003) consisting only of a phrase table and trigram language model. Under this simple model, for a French sentence f of length I, English sentence e of length J, and alignment a where each element consists of a span in both e and f such that every word in both e and f is aligned exactly once, the conditional probability of e and a given f is as follows.[4]

$$p(e, a|f) \propto \prod_{\langle i,i',j,j'\rangle \in a} p(f_i^{i'}|e_j^{j'}) \prod_{j=1}^{J+1} p(e_j|e_{j-1}, e_{j-2}) \quad (1)$$

To evaluate output, we compute the conditional probability of e as follows.

[2] `http://www.isi.edu/natural-language/download/hansard/`

[3] `http://mt-class.org/jhu/hw2.html`

[4] For simplicity, this formula assumes that e is padded with two sentence-initial symbols and one sentence-final symbol, and ignores the probability of sentence segmentation, which we take to be uniform.

$$p(e|f) \propto \sum_{a} p(e, a|f) \tag{2}$$

Note that this formulation is different from the typical Viterbi objective of standard beam search decoders, which do not sum over all alignments, but approximate $p(e|f)$ by $\max_a p(e, a|f)$. Though the computation in Equation 2 is intractable (DeNero and Klein, 2008), it can be computed in a few minutes via dynamic programming on reasonably short sentences, a criterion met by the 48 sentences we chose from the Canadian Hansards. The corpus-level probability is then the product of all sentence-level probabilities in the data. Since this computation takes more than a few seconds, we added functionality to the leaderboard to display a progress bar, which can be reused by other custom scorers following the methods used in `scoring/decode.py`. The corresponding datasets are in `scoring/decoding_data`. To score against a different dataset, simply change the data files.

3.2.3. Assignment 3: Evaluate

In this assignment, the input is a dataset in which each sample consists of a reference sentence and a pair of translation outputs. The task is to decide which of the translation outputs is better, or if they are of equal quality.[5] Hence the task is a three-way classification problem for each input, optionally using the reference to compute features (which might include standard evaluation measures such as BLEU). To evaluate, results are compared against human assessments of the translation pairs, taken from the 2012 Workshop on Machine Translation (Callison-Burch et al., 2012). In our homework assignments, we also provided a training dataset for which human assessments are provided so that students can train classifiers for the problem. The scorer is implemented in `scoring/evaluation.py` with data in `scoring/eval_data`. To score against a different dataset, simply change the data files.

3.2.4. Assignment 4: Rerank

In this assignment, the input consists of n-best lists of translations and their associated features produced by a machine translation system on a test corpus. The task is to select the best translation for each input sentence according to BLEU, computed against a hidden reference sentence. The n-best lists were provided by Chris Dyer as an entry for the Russian-English translation task in the 2013 Workshop on Machine Translation (Bojar et al., 2013). The scorer is implemented in `scoring/rerank.py`, and the corresponding datasets are in `scoring/rerank_data`. To score against a different dataset, simply change the data files.

[5] `http://mt-class.org/jhu/hw3.html`

Table 2. Assignment five (inflection) statistics.

split	sentences	tokens	lemmas
train	29,768	518,647	35,701
dev	4,042	70,974	11,304
test	4,672	80,923	11,655

3.2.5. Assignment 5: Inflect

In this assignment, the input is a sequence of lemmatized Czech words. The task is to choose the correct inflection for each word (reminiscent of Minkov et al. (2007)). For example:

(1) Oba tyto úkoly jsou vědecky i technicky mimořádně obtížné .
oba‘2 tento úkol být vědecky_(*1ý) i-1 technicky_(*1ý) mimořádně_(*1ý) obtížný .
‘Both of these tasks are scientifically and technically extremely difficult.’

The data comes from the Prague Dependency Treebank v2.0, which is distributed through the Linguistic Data Consortium.[6] The homework assignment[7] contains instructions and a script to format the data directly from the LDC repository directory.

3.3. Creating New Assignments

The MT Leaderboard is easily extended with new assignments:

1. Create an entry for the assignment in list `scorer` near the top of `leaderboard.py`:

```
scorer = [
  scoring.upload_number,
  scoring.new_assignment,
  # scoring.alignment,
  # scoring.decode,
  # scoring.evaluation,
  # scoring.rerank,
  # scoring.inflect,
]
```

2. Next, create a file `scoring/new_assignment.py`. This file must define four variables (the assignment name, a text description of its scoring method for the

[6] `https://catalog.ldc.upenn.edu/LDC2006T01`

[7] `http://mt-class.org/jhu/hw5.html`

leaderboard header, a boolean indicating the scoring method sort order, and the assignment deadline) and two functions, `score(...)` and `oracle()`

```
$ cd scoring
$ cp upload_number.py new_assignment.py
# Edit new_assignment.py
```

3. Place any data in the directory `scoring/new_assignment_data/`.

That's it. Assignments become available as soon as they are listed in the main `leaderboard.py` script, and students can upload assignments as long as the deadline hasn't passed.

4. Case Study

In our spring 2014 class at Johns Hopkins, we received 307 submissions from 17 students over five assignments using the leaderboard as described here. Students responded positively to the leaderboard, for instance commenting that "The immediate feedback of the automatic grading was really nice". Some students used the leaderboard grader for a large number of experiments, which they then reported in writeups. For further information on how we incorporated the leaderboard into our class, empirical results, and student responses, see Lopez et al. (2013).

The MTL is only one component of a good class on machine translation. The time we saved was put into other tasks, including an emphasis on scientific writing: Students were required to submit a thorough ACL-style writeup of every homework assignment, including a description of the problem, a description of their approach, and quantitative and qualitative analysis of their findings. These writeups were graded carefully, and students received feedback on their writing. We also required students to submit their code, which we manually reviewed.

5. Future Work

With the display of a leaderboard sorted by student scores against hidden development data, the MT Leaderboard provides a competitive environment in hopes of motivating students to experiment with different approaches. However, competition isn't always the best motivator; some of our students chose to hide their handles from the leaderboard. We considered but did not implement a more cooperative approach in which students work together to improve an oracle computed over all of their submissions. For example, in the alignment setting, we could select, for each sentence, the one with the best AER across all students, and then compute AER over the whole set. At submission time, the student could then be shown how much their submission increased the oracle score. This idea could also be extended to system combination, for example, by having student submissions vote on alignment links.

Acknowledgements

We thank Chris Dyer for improving our assignments, and the students of our 2014 class at Johns Hopkins for testing everything out.

Bibliography

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 workshop on statistical machine translation. In *Proc. of WMT*, 2013.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proc. of WMT*, 2012.

DeNero, John and Dan Klein. The complexity of phrase alignment problems. In *Proc. of ACL*, 2008.

Hajič, Jan, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. Prague Dependency Treebank 2.0. LDC2006T01, Linguistic Data Consortium, Philadelphia, PA, USA, ISBN 1-58563-370-4, Jul 2006, 2006. URL `http://ufal.mff.cuni.cz/pdt2.0/`.

Koehn, Philipp, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of NAACL*, 2003.

Lopez, Adam, Matt Post, Chris Callison-Burch, , Jonathan Weese, Juri Ganitkevitch, Narges Ahmidi, Olivia Buzek, Leah Hanson, Beenish Jamil, Matthias Lee, Ya-Ting Lin, Henry Pao, Fatima Rivera, Leili Shahriyari, Debu Sinha, Adam Teichert, Stephen Wampler, Michael Weinberger, Daguang Xu, Lin Yang, and Shang Zhao. Learning to translate with products of novices: a suite of open-ended challenge problems for teaching MT. *Transactions of the Association for Computational Linguistics*, (1):165–178, 2013.

Mihalcea, Rada and Ted Pedersen. An evaluation exercise for word alignment. In *Proc. on Workshop on Building and Using Parallel Texts*, 2003.

Minkov, Einat, Kristina Toutanova, and Hisami Suzuki. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, pages 128–135, 2007.

Och, Franz Josef and Hermann Ney. Improved statistical alignment models. In *Proc. of ACL*, 2000.

Address for correspondence:
Matt Post
`post@cs.jhu.edu`
Human Language Technology Center of Excellence
Johns Hopkins University
810 Wyman Park Drive
Baltimore, MD 21211

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 47–56

Depfix, a Tool for Automatic Rule-based Post-editing of SMT

Rudolf Rosa

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

We present Depfix, an open-source system for automatic post-editing of phrase-based machine translation outputs. Depfix employs a range of natural language processing tools to obtain analyses of the input sentences, and uses a set of rules to correct common or serious errors in machine translation outputs. Depfix is currently implemented only for English-to-Czech translation direction, but extending it to other languages is planned.

1. Introduction

Depfix is an automatic post-editing system, designed for correcting errors in outputs of English-to-Czech statistical machine translation (SMT) systems. An approach based on similar ideas was first used by Stymne and Ahrenberg (2010) for English-to-Swedish. Depfix was introduced in (Mareček et al., 2011), and subsequent improvements were described especially in (Rosa et al., 2012b) and (Rosa et al., 2013). For a comprehensive description of the whole Depfix system, please refer to (Rosa, 2013). An independent implementation for English-to-Persian exists, called Grafix (Mohaghegh et al., 2013).

Depfix consists of a set of rule-based fixes, and a statistical component.[1] It utilizes a range of NLP tools, especially for linguistic analysis of the input (taggers, parsers, named entity recognizers…), and for generation of the output (morphological generator, detokenizer…). Depfix operates by analyzing the input sentence, and invoking a pipeline of error detection and correction rules (called *fixes*) on it.

Depfix is one of the components of Chimera (Bojar et al., 2013b; Tamchyna et al., 2014), the current state-of-the-art system for English-to-Czech machine translation

[1] However, the vital part of Depfix are the rule-based fixes.

 Corresponding author: rosa@ufal.mff.cuni.cz
Cite as: Rudolf Rosa. Depfix, a Tool for Automatic Rule-based Post-editing of SMT. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 47–56. doi: 10.2478/pralin-2014-0013.

System	WMT 2011	WMT 2012	System	WMT 2011	WMT 2012
CU Bojar	+**0.47**	+0.07	JHU	+**0.42**	+**0.32**
CU Tamchyna	+**0.46**	+0.02	SFU	–	+**0.41**
CU TectoMT	−**0.10**	−0.02	EuroTran	+**0.21**	+**0.15**
CU Zeman	+**0.73**	+**0.34**	Microsoft Bing	–	+**0.37**
UEDIN	+**0.64**	+**0.23**	Google Translate	+**0.23**	0.00

Table 1. Automatic evaluation of the Depfix system. Adapted from (Rosa, 2013). Change of BLEU score when Depfix was applied to the output of the system is reported. Statistically significant results are marked by bold font.

(MT) – the other components are TectoMT (Žabokrtský et al., 2008) and factored Moses (Koehn et al., 2007). Chimera has ranked as the best English-to-Czech MT system in the last two translation tasks of the Workshop on Statistical Machine Translation (WMT) (Bojar et al., 2013a, 2014). Depfix is currently being developed in the frame of QTLeap,[2] a project focusing on quality translation by deep language engineering approaches.

Depfix is a stand-alone system, and can be used to post-process outputs of any MT system. It particularly focuses on errors common in phrase-based SMT outputs; some of its components have been tuned using outputs of Moses. In a throughout evaluation on outputs of all systems participating in WMT in 2011 and 2012 (Callison-Burch et al., 2011, 2012), applying Depfix led to a statistically significant improvement in BLEU score in most cases, as shown in Table 1.[3]

Depfix is implemented in the Treex framework (Popel and Žabokrtský, 2010).[4] Instructions on obtaining and using Depfix can be found on `http://ufal.mff.cuni.cz/depfix`. We release Depfix under the GNU General Public License v2 to encourage its improvement and adaptation for other languages, as well as to serve as inspiration for other researchers.

2. Tools

Depfix basically operates by observing and modifying morphological tags. Therefore, the two following tools are vital for operation of Depfix:

- A **lemmatizing tagger** (or a tagger and a lemmatizer) is an analysis tool that assigns the word form of each token in the sentence with a combination of lemma

[2] `http://qtleap.eu/`

[3] We did not perform this kind of evaluation in the following years of WMT, as we focused on the Chimera hybrid system instead.

[4] `http://ufal.mff.cuni.cz/treex`

and tag. We use MorphoDiTa (Straková et al., 2014) for Czech and Morče (Spoustová et al., 2007) for English.
- A **morphological generator** is a generation tool inverse to the tagger: for a given combination of lemma and tag, it generates the corresponding word form. We use Hajič's Czech morphological generator (Hajič, 2004).

For Czech, we use the Prague dependency treebank positional tagset (Hajič, 1998), which marks 13 morphological categories, such as part-of-speech, gender, number, case, person, or tense. One of the properties of this tagset, which is very useful for us, is that the lemmas and morphological categories are fully disambiguated – for a given combination of lemma and tag, there is at most one corresponding word form (the opposite does not hold, as many Czech paradigms have the same word repeated several times).

For English, we use the Penn treebank tagset (Marcus et al., 1993), which marks only few morphological categories, such as singular/plural number for nouns, but does not distinguish e.g. verb person (except for 3rd person singular in present simple tense).

Apart from the tagger and the morphological generator, many other tools are used in Depfix. We currently use the following, which are either implemented within the Treex framework, or are external tools with Treex wrappers:
- a rule-based Treex tokenizer and detokenizer
- a word aligner – GIZA++ (Och and Ney, 2003)
- a dependency parser – MST parser for English (McDonald et al., 2005), and its variations for Czech: a version by Novák and Žabokrtský (2007) adapted for Czech in the basic version of Depfix, or MSTperl by Rosa et al. (2012a) adapted for SMT outputs in full Depfix
- a dependency relations labeller (as the MST parser returns unlabelled parse trees) – a rule-based Treex labeller for English, and a statistical labeller by Rosa and Mareček (2012) for Czech
- a named entity recognizer – Stanford NER for English (Finkel et al., 2005), and a simple Treex NER for Czech
- a rule-based Treex converter to tectogrammatical (deep syntax) dependency trees

There are also other tools that we do not currently use (because they are not part of Treex yet, some of them probably do not even exist yet), but we believe that they would be useful for Depfix as well:
- a full-fledged named entity recognizer for Czech
- a coreference resolver
- a fine-grained tagger for English (that would mark e.g. verb person or noun gender)
- a well-performing labeller for tectogrammatical trees (the current Treex one is rather basic and lacks proper analysis of verbs, negation, etc., especially for English)

When porting Depfix to a new language, acquiring the NLP tools is a necessary first step. Unfortunately, in the Treex framework, support for languages other than Czech and English is currently very limited. However, one can make use of the HamleDT project (Zeman et al., 2012),[5] which collects dependency treebanks for various languages and harmonizes their tagsets and dependency annotation styles to a common scheme. We believe this to be an ideal resource for training a tagger as well as a dependency parser for any of the languages covered – HamleDT 2.0 currently features 30 treebanks and is still growing, and there are also plans of its tighter integration with Treex.

3. Fixing Rules

The main part of Depfix is a set of fixing rules (there are 28 of them in the current version). Most of the rules inspect the tag of a Czech word, usually comparing it to its source English counterpart and/or its Czech neighbours (usually its dependency parents or children), and if an error is spotted (such as incorrect morphological number – e.g. the Czech word is in singular but the source English word is in plural), the tag of the Czech word is changed to the correct one, and the morphological generator is invoked to generate the corresponding correct word form.[6]

Some of the rules also delete superfluous words (e.g. a subject pronoun that should be dropped), change word order (e.g. the noun modifier of a noun, which precedes the head noun in English but should follow it in Czech), or change tokenization and casing (by projecting it from the source English sentence where this seems appropriate).

The ideas for the rules are based on an analysis of errors in English-to-Czech SMT (Bojar, 2011), and the actual rules were implemented and tuned using the WMT 2010 test set (Callison-Burch et al., 2010) translated by Moses. For other language pairs, a similar error analysis, such as the Terra collection (Fishel et al., 2012), may be used as a starting point; however, the error analyses are typically not fine-grained enough to be used directly for Depfix rules implementation, and extensive manual tuning of the rules by inspecting SMT translation outputs is to be expected.

3.1. Example

Table 2 shows the operation of *FixPnom* rule.[7] In the sentence, there is an error in agreement of nominal predicate "zdrženliví" ("reticent$_{\mathtt{pl}}$") with the subject "Obama".

[5] `http://ufal.mff.cuni.cz/hamledt`

[6] It may happen that the new word form turns out to be identical to the original word form, as there are often many possible tags for a word – e.g. the nominative and accusative case of a noun is often identical. However, the fix may still be beneficial, as subsequent fixes might be helped by the corrected tag of the word – e.g. a fixed noun tag may induce a fix of a modifying adjective.

[7] The **`make`** `compare_log` Depfix command can be used to obtain this kind of information.

Source:	Obama has always been reticent in regards to his prize.
SMT output:	Obama byl vždy **zdrženliví** s ohledem na svou kořist.
Depfix output:	Obama byl vždy **zdrženlivý** s ohledem na svou kořist.
Fixlog:	Pnom: zdrženliví[AAM**P**1—-1A—-] zdrženlivý[AAM**S**1—-1A—-]

Table 2. Example of application of FixPnom *on a sentence from WMT10 dataset (translated by the CU-Bojar system)*

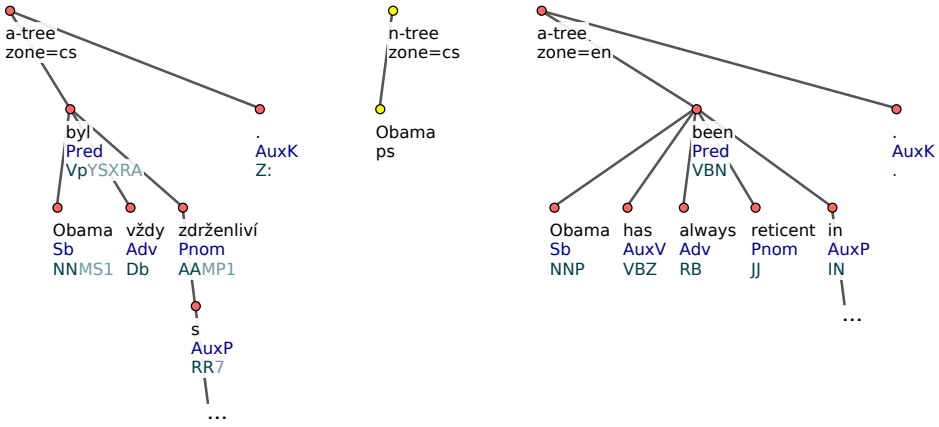


Figure 1. Part of the dependency parse tree of a Czech sentence before applying FixPnom*. Also showing the Czech named entity tree, and corresponding part of the source English dependency parse tree. Word forms, analytical functions, and morphological tags are shown.*

The morphological number (4th position of the tag) should be identical for both of the words, but it is not – it is singular ("S") for "Obama" but plural ("P") for "zdrženliví". See also Figure 1, which shows the parse tree of the Czech sentence (before applying the fix), the named entity tree for the Czech sentence, and the parse tree of the source English sentence.[8]

When the *FixPnom* rule is invoked on the word "zdrženliví", it realizes the following:

- the word is an adjective and its dependency parent is a copula verb, thus the word is a nominal predicate and the *FixPnom* rule applies here,

[8]These parse trees, as well as the tectogrammatical trees, are contained in intermediate `*.treex` files in the Depfix experiment directory, and can be viewed using Tree Editor TrEd – see `http://ufal.mff.cuni.cz/tred/`.

- there is a child of the parent verb ("Obama") which is marked as subject, and its English counterpart ("Obama") is also marked as subject, thus it should be in agreement with the nominal predicate,
- the subject is in singular, while the nominal predicate is in plural, thus the agreement is violated and should be fixed.

The rule therefore proceeds by fixing the error. This is done in two steps:

1. the tag of "zdrženliví" is changed by changing the number marker from "P" (plural) to "S" (singular), as indicated in the Fixlog in Table 2,[9]
2. the morphological generator is invoked to generate a word form that corresponds to the new tag; in this case, the word "zdrženlivý" is generated.

The *FixPnom* rule also checks and corrects agreement in morphological gender; however, agreement in gender is not violated in the example sentence.

4. Implementation

Depfix is implemented in the Treex framework, which is required to run it, and is operated from the command-line via Makefile targets. The commented source code of Depfix is in Perl and Bash. The fixing blocks are implemented as Treex blocks, usually taking a dependency edge as their input, checking it for the error that they fix, and fixing the child or parent node of the edge as appropriate.

The Depfix Manual (Rosa, 2014a), which provides instructions on installing and running Depfix, is available on the Depfix webpage. The installation consists of installing Treex and several other modules from CPAN, checking out the Treex subversion repository (which Depfix is contained in), downloading several model files, and making a test run of Depfix.

Depfix needs a Linux machine to run, with at least 3.5 GB RAM to run the basic version – i.e. without the MSTperl parser, which is adapted for SMT outputs (Rosa et al., 2012a; Rosa, 2014b), and without the statistical fixing component (Rosa et al., 2013). The full version, which achieves slightly higher BLEU improvement than the basic version, needs at least 20 GB to run.

Depfix takes source English text and its machine translation as its input, and provides the fixed translations as its output (all plain text files, one sentence per line). Processing a set of 3000 sentences by Depfix takes about 2 hours; processing a single sentence takes about 5 minutes (most of this time is spent by initializing the tools).[10]

[9] The fix is performed in this direction because the morphological number is more reliable with nouns in English-to-Czech translation, as noun number is explicitly marked in English while adjective number is not.

[10] These times are provided for illustration only, as they depend on the speed of the processor, the hard-drive, and other parameters of the machine.

5. Conclusion and Future Work

In this paper, we described Depfix, a successful automatic post-editing system system designed for performing rule-based correction of errors in English-to-Czech statistical machine translation outputs.

As a stand-alone tool, Depfix can be used to post-edit outputs of any machine translation system, although it focuses especially on shortcomings of the phrase-based ones, such as Moses. So far, we have implemented Depfix only for the English-to-Czech translation direction, although there exist similar systems for other languages by other authors. Depfix has been developed for several years, and is now a component of Chimera, the state-of-the-art machine translation system for English-to-Czech translation.

The future plans for Depfix development are directed towards extending it to new translation directions, starting with a refactoring to separate language-independent and language-specific parts, so that fixing rules for a new language pair can be implemented easily while reusing as much from the already implemented functionality as possible. Another future research path aims to complement or replace the manually written rules by machine learning techniques, with preliminary experiments indicating viability of such an approach.

To improve the ease of use of Depfix, we also wish to implement an online interface that would enable invoking Depfix remotely from the web browser or as a web service, with no need of installing it. The interface will be implemented using the Treex::Web front-end (Sedlák, 2014).[11]

Acknowledgements

This research was supported by the grants FP7-ICT-2011-7-288487 (MosesCore), FP7-ICT-2013-10-610516 (QTLeap), GAUK 1572314, and SVV 260 104. This work has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013).

Bibliography

Bojar, Ondřej. Analyzing Error Types in English-Czech Machine Translation. *Prague Bulletin of Mathematical Linguistics*, 95:63–76, March 2011. ISSN 0032-6585.

Bojar, Ondrej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, 2013a.

[11] `https://ufal.mff.cuni.cz/tools/treex-web`

Bojar, Ondřej, Rudolf Rosa, and Aleš Tamchyna. Chimera – three heads for English-to-Czech translation. In *Proceedings of the Eight Workshop on Statistical Machine Translation*, pages 92–98, Sofija, Bulgaria, 2013b. Bălgarska akademija na naukite, Association for Computational Linguistics.

Bojar, Ondrej, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.statmt.org/wmt10/pdf/WMT03.pdf`.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omar Zaidan. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics.

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

Fishel, Mark, Ondrej Bojar, and Maja Popovic. Terra: a collection of translation error-annotated corpora. In *LREC*, pages 7–14, 2012.

Hajič, Jan. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, 2004.

Hajič, Jan. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In Hajičová, Eva, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press, 1998.

Koehn, Philipp, 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. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Comp. Ling.*, 19:313–330, June 1993. ISSN 0891-2017.

Mareček, David, Rudolf Rosa, Petra Galuščáková, and Ondřej Bojar. Two-step translation with grammatical post-processing. In Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omar Zaidan, editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 426–432, Edinburgh, UK, 2011. University of Edinburgh, Association for Computational Linguistics.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics, 2005.

Mohaghegh, Mahsa, Abdolhossein Sarrafzadeh, and Mehdi Mohammadi. A three-layer architecture for automatic post-editing system using rule-based paradigm. *WSSANLP-2013*, page 17, 2013.

Novák, Václav and Zdeněk Žabokrtský. Feature engineering in maximum spanning tree dependency parser. In Matoušek, Václav and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, Lecture Notes in Computer Science, pages 92–98, Pilsen, Czech Republic, 2007. Springer Science+Business Media Deutschland GmbH.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

Popel, Martin and Zdeněk Žabokrtský. TectoMT: modular NLP framework. In *Proceedings of the 7th international conference on Advances in natural language processing*, IceTAL'10, pages 293–304, Berlin, Heidelberg, 2010. Springer-Verlag.

Rosa, Rudolf. Automatic post-editing of phrase-based machine translation outputs. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2013. URL `http://ufal.mff.cuni.cz/rudolf-rosa/master-thesis`.

Rosa, Rudolf. Depfix manual. Technical Report TR-2014-55, ÚFAL MFF UK, 2014a. URL `http://ufal.mff.cuni.cz/techrep/tr55.pdf`.

Rosa, Rudolf. MSTperl parser, 2014b. URL `http://hdl.handle.net/11858/00-097C-0000-0023-7AEB-4`.

Rosa, Rudolf and David Mareček. Dependency relations labeller for Czech. In Sojka, Petr, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 15th International Conference, TSD 2012. Proceedings*, number 7499 in Lecture Notes in Computer Science, pages 256–263, Berlin / Heidelberg, 2012. Masarykova univerzita v Brně, Springer Verlag.

Rosa, Rudolf, Ondřej Dušek, David Mareček, and Martin Popel. Using parallel features in parsing of machine-translated sentences for correction of grammatical errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea, 2012a. Association for Computational Linguistics.

Rosa, Rudolf, David Mareček, and Ondřej Dušek. DEPFIX: A system for automatic correction of Czech MT outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, 2012b. Association for Computational Linguistics.

Rosa, Rudolf, David Mareček, and Aleš Tamchyna. Deepfix: Statistical post-editing of statistical machine translation using deep syntactic analysis. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 172–179,

Sofija, Bulgaria, 2013. Bălgarska akademija na naukite, Association for Computational Linguistics.

Sedlák, Michal. *Treex::Web*. Bachelor's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague, Czechia, 2014. URL `https://lindat.mff.cuni.cz/services/treex-web/`.

Spoustová, Drahomíra, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. The best of two worlds: Cooperation of statistical and rule-based taggers for Czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing, ACL 2007*, pages 67–74, Praha, 2007.

Straková, Jana, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Stymne, Sara and Lars Ahrenberg. Using a grammar checker for evaluation and postprocessing of statistical machine translation. In *LREC*, 2010.

Tamchyna, Aleš, Martin Popel, Rudolf Rosa, and Ondřej Bojar. CUNI in WMT14: Chimera still awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 195–200, Baltimore, MD, USA, 2014. Association for Computational Linguistics.

Žabokrtský, Zdeněk, Jan Ptáček, and Petr Pajas. TectoMT: Highly modular MT system with tectogrammatics used as transfer layer. In *ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170, Columbus, OH, USA, 2008. Association for Computational Linguistics.

Zeman, Daniel, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To parse or not to parse? In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

Address for correspondence:
Rudolf Rosa
`rosa@ufal.mff.cuni.cz`
Charles University in Prague,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 57–68

A Set of Annotation Interfaces for Alignment of Parallel Corpora

Anil Kumar Singh

IIT (BHU), Varanasi, India

Abstract

Annotation interfaces for parallel corpora which fit in well with other tools can be very useful. We describe a set of annotation interfaces which fulfill this criterion. This set includes a sentence alignment interface, two different word or word group alignment interfaces and an initial version of a parallel syntactic annotation alignment interface. These tools can be used for manual alignment, or they can be used to correct automatic alignments. Manual alignment can be performed in combination with certain kinds of linguistic annotation. Most of these interfaces use a representation called the Shakti Standard Format that has been found to be very robust and has been used for large and successful projects. It ties together the different interfaces, so that the data created by them is portable across all tools which support this representation. The existence of a query language for data stored in this representation makes it possible to build tools that allow easy search and modification of annotated parallel data.

1. Introduction

Machine translation, at least for certain language pairs, has reached a point where it is now being practically used by professional translators as well as users. Many, perhaps a majority of these machine translation systems are based on the statistical approach (Koehn et al., 2007). The general architecture of these machine translation systems is easily portable to other language pairs than those for which they were made. However, the one major drawback of these systems is that they need a large quantity of aligned parallel text. While the rule-based approach (Corbí-Bellot et al., 2005) may provide a feasible solution in many cases where such corpora are lacking, the other alternative, i.e., creating the needed parallel corpora for language pairs that lack them can still be an effective solution under certain conditions such as the availability of sufficient quantity of parallel text in electronic form. Still, even when such

 Corresponding author: nlprnd@gmail.com
Cite as: Anil Kumar Singh. A Set of Annotation Interfaces for Alignment of Parallel Corpora. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 57–68. doi: 10.2478/pralin-2014-0014.

text is available, there is a lot of work that needs to be done if the text is not already sentence aligned. Therefore, sentence alignment tools are one of the first enablers for creating machine translation systems based on the statistical approach in cases where sentence alignment accuracy is not close to 100%. If the text can be further aligned at the word level, it becomes a valuable resource for many other purposes.

Tools for automatic alignment of text, both at the sentence level (Brown et al., 1991; Gale and Church, 1991) and at the word level (Gale and Church, 1993; Och and Ney, 2003) are available. However, word alignment tools do not have an accuracy that will allow them to be used directly without further correction, except as part of a statistical machine translation system. Sentence alignment tools reportedly have very good accuracies, but on closer inspection we find that they do break down under certain conditions which are especially likely to occur for language pairs which lack sentence aligned parallel corpora. The languages should not be very different in phylogenetic terms and, more importantly, the text should not be 'noisy' (Singh and Husain, 2005), which practically means that it should be almost aligned already.

Beyond the word level, we could also have corpora which are partially aligned for specific grammatical elements such as nominal compounds, or even corpora which are syntactically aligned, e.g. parallel treebanks (Li et al., 2012). Such corpora are even more valuable for not only linguistic analysis and extraction of linguistic patterns but also for machine learning of Natural Language Processing (NLP) tasks.

2. The Workflow

The above are the reasons why we need annotation interfaces for parallel corpora. We need a set of interfaces that enable the complete workflow from sentence alignment to word alignment to syntactic (or even semantic) alignment. There are not many such interfaces available, at least in the open source domain. One that is available is called Uplug[1] (Tiedemann, 2003). It allows sentence alignment, word alignment and with some extensions such as in UplugConnector[2], even treebank alignment.

In the following paragraphs we describe a set of interfaces that aim to implement the complete process of parallel corpora alignment up to the syntactic (treebank) level. All these interfaces are part of the same suite of tools and APIs called Sanchay[3].

The workflow starts with some parallel text that is not sentence aligned, but is tokenized into sentences and words. An automatic sentence alignment tool can optionally be run on this parallel text. The text is then fed into the sentence alignment interface, where a user manually corrects the alignments marked by the sentence alignment tool. Alternatively, the user can directly mark the alignments completely manually. The output of the sentence alignment interface can then be run through an

[1] http://sourceforge.net/projects/uplug/

[2] http://www2.lingfil.uu.se/personal/bengt/uconn113.html

[3] http://sanchay.co.in

automatic word alignment tool, and then to the word alignment interface. A user then corrects the word alignments errors. The user can also start from scratch at the word alignment level and mark all the alignments manually. It is possible at this stage to create a shallow parsed aligned parallel corpus. Such a corpus will have chunks or word groups marked, (optionally) tagged and aligned.

The output of the word alignment interface can be run through an automatic syntactic analysis tool such as a parser. If the goal is ambitious, it may be possible to create parallel treebanks using the syntactic annotation interface available in the same suite. This interface is meant for creating syntactico-semantic resources such as treebanks.

Finally, there is a parallel syntactic annotation tool that will allow users to align the parallel treebanks at the deep syntactic level. Aligned parallel treebanks can allow a wealth of information to be extracted from them.

3. The Representation

For all the interfaces mentioned here except one, the data is stored in memory as well in files using a representation called the Shakti Standard Format or SSF (Bharati et al., 2014). This representation is a robust way of storing data which is the result of linguistic analysis, particularly syntactico-semantic analysis.

An example sentence in SSF is show below:

```
Address Token       Category       Attribute-value pairs
  ------------------------------------------------
  1     ((          NP
  1.1   children    NNS  <fs af=child,n,m,p,3,0,,>
        ))
  2     ((          VG
  2.1   are         VBP  <fs af=be,v,m,p,3,0,,>
  2.2   watching    VBG  <fs af='watch,v,m,s,3,0,,' aspect=PROG>
        ))
  3     ((          NP
  3.1   some        DT   <fs af=some,det,m,s,3,0,,>
  3.2   programmes  NNS  <fs af=programme,n,m,p,3,0,,>
        ))
  4     ((          PP
  4.1   on          IN   <fs af=on,p,m,s,3,0,,>
  4.1.1 ((          NP
  4.1.2 television  NN   <fs af=television,n,m,s,3,0,,>
        ))
        ))
  ------------------------------------------------
             Shakti Standard Format
```

In this representation, sentences are the nodes of a tree at the document level and the constituents are the nodes at the sentence level. Each node has a possible lexical

item, a tag (such as a part-of-speech or POS tag or a phrase tag) and a feature structure associated with it. SSF allows not only features of nodes to be stored, but also features across nodes, which represent relationships across words, chunks, phrases, nodes of a dependency tree or even sentences and documents. We have used an attribute called 'alignedTo' for storing the alignments, whether of words, chunks, wordgroups, phrases or of sentences or documents. This attribute takes a string value and multiple alignments can be given by using semi-colon as the separator.

Figures 1 and 2 show the same sentence analyzed according to phrase structure grammar and dependency grammar, respectively. SSF can encode both of them in the same place. For example, the underlying tree can be a phrase structure tree and the dependency tree can be encoded over the phrase structure tree using an attribute like 'drel' (dependency relation) and 'name' (a unique identifier for the node). The value of the 'drel' attribute is in two parts, separated by a colon, e.g. 'k1:children'. The first part is the dependency relation and the second part is the unique name.

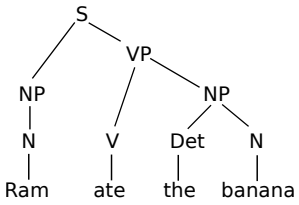


Figure 1. Phrase structure tree

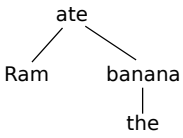


Figure 2. Dependency tree

We have opted for SSF because it allows us to preserve many different kinds of linguistic information about the text being aligned. The alignment information is added as an extra layer in the form of the 'alignedTo' attribute. Moreover, since we are storing alignments on both sides, the interfaces have to ensure that the values of the 'alignedTo' attribute are kept synchronized on the two sides. This makes it possible to get the alignments from either side to the other side. SSF allows us to keep all the information in one place and to have the data in a readable form so that even without a visualizer it can still be made sense of by a human user.

4. Sentence Alignment Interface

The sentence alignment interface (Figure 3) takes as input the text which has been tokenized into sentences. It shows one sentence per row in each table: one table on the source side and one on the target side. It includes an implementation of a sentence alignment algorithm which is based on sentence lengths. Such algorithms are based on the intuition that the lengths of translations are likely to be roughly proportional to lengths of sentences in the source language. The length can be calculated in terms

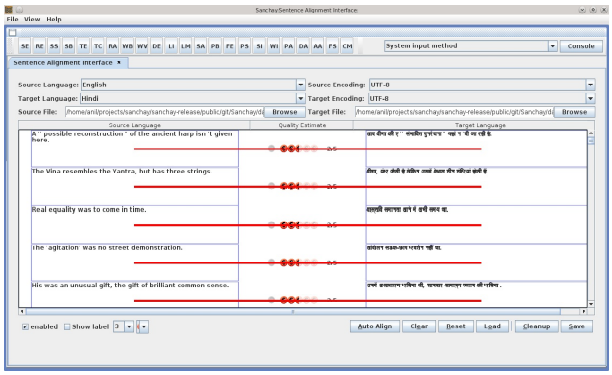


Figure 3. Sentence Alignment Interface: Initial alignments have been marked automatically

of words (Brown et al., 1991) or in terms of characters (Gale and Church, 1991). The latter has been shown to work better, perhaps because more data is available. In our implementation, we use both of them as features with weights. The alignment algorithm is a dynamic programming based algorithm. This implementation is available as the default, but with some minimal effort it is possible to plug-in any other available implementation of some sentence alignment algorithm, such as that in (Moore, 2002), which has been shown to be quite robust (Singh and Husain, 2007).

The user can mark manual alignments or correct automatic alignments by the simple mechanism of drag-and-drop. Deleting an alignment also uses this mechanism: If an alignment already exists, drag-and-drop deletes it. Multiple alignments to the same sentence are allowed on either sides. The interface tries to keep the sentences displayed according to the alignments, i.e., the display changes based on the alignments. This ensures that the user does not have to drag-and-drop too far because the potential alignment is as close to the source language sentence as possible.

The interface design is based on the assumption that sometimes the input to the interface can come from a machine translation system. In such a case, it is possible to extend the interface to allow the annotation of quality estimation measurement as well as to allow it to be used as a post-editing tool. Some work has already been done in this direction. Completing it is one of the future directions of this work.

The first step in using all the interfaces is to start Sanchay (see `http://sanchay.co.in`). Then the respective interfaces can be opened either by clicking on the buttons in the toolbox or from the 'File' menu. The toolbox buttons have two letter symbols for each interface. For example, the symbol for the word alignment interface is 'WI'. These could perhaps better be replaced by suitable icons. Hovering the cursor on the symbols displays their full name. Clicking on 'WI' will open the word alignment in-

terface. The symbol for the sentence alignment interface is 'SI', for the parallel corpus markup interface it is 'PA', for the syntactic annotation interface it is 'SA' and for the parallel syntactic annotation alignment interface it is 'PS'.

To start the alignment process, the user has to first browse to the source and the target files, where the target file should be the translation of the source file. Then clicking on the button 'Load' opens these files and displays them in the interface. The 'Auto Align' button is for running the automatic sentence aligner on the opened files, so that the work is reduced and only automatic alignment errors need to be corrected. Now the user can start marking the alignments by drag-and-drop.

The output is in the form of two files: source and target. Both files have sentences in the Shakti Standard Format. Each aligned sentence has an attribute 'alignedTo', whose value points to the unique id of the aligned sentence in the other file:

```
<Sentence id='1' alignedTo='1'>
```

5. Word Alignment Interface

Since the data representation (SSF) is common, the word alignment interface (Figure 4) can directly take in the output of the sentence alignment interface. This interface is actually more than a word alignment interface. It allows words to be grouped together on both sides, to be tagged, and then allows these groups to be aligned. Just like in the sentence alignment interface, a drag-and-drop over an existing alignment deletes that alignment.

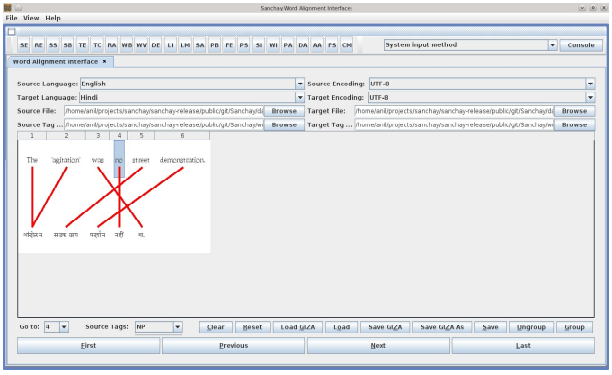


Figure 4. Word Alignment Interface

There is an implementation of a word alignment algorithm based on the first three IBM models (Brown et al., 1993), but that is not yet connected to the interface (another possible future direction for this work). However, the interface can read the output of

the most popular word alignment tool, namely GIZA++[4]. Therefore, it is possible to use this interface to correct the output of GIZA++. It can also save data in the same (GIZA++) format for those who are not comfortable with SSF or because the user wants to further process the corrected alignments in this format. This connectivity with GIZA++ makes the tool more useful.

While the sentence alignment interface uses the 'alignedTo' attribute at the sentence level, this interface uses the same attribute at the word or the chunk level to mark the alignments in the data representation.

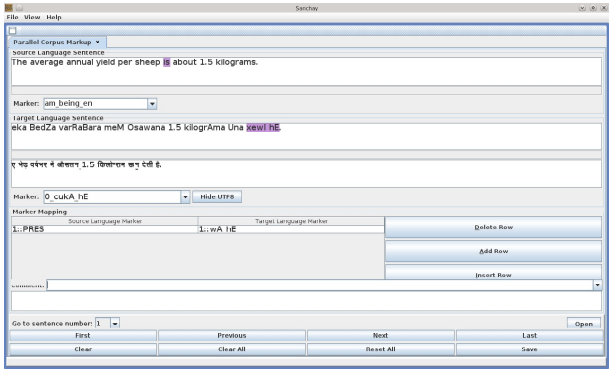


Figure 5. Parallel Markup Interface

The user can start the alignment process in two ways. One is by loading the source and target files (whether containing automatic alignments or not) and start working on them, either from scratch or by correcting errors. The other is to load the output of GIZA++ (in '*.A3.*' format) into the interface and correct the errors in automatic alignment. Since the interface allows grouping and tagging also, the source and target tag files may also need to be loaded.

The output from the interface is again in the form of two files, one source and one target, and both files have sentences in the Shakti Standard Format. Each aligned node has an attribute 'alignedTo', whose value points to the unique name of the aligned node in the corresponding sentence in the other file:

```
<Sentence id='1' alignedTo='1'>
...
8       ancient         <fs name='ancient' alignedTo='prAcIna'>
9       flute           <fs name='flute' alignedTo='vInA'>
...
```

[4] https://code.google.com/p/giza-pp

6. Parallel Markup Interface

Unlike the other interfaces in the suite, this one (Figure 5) does not use SSF. Instead, it uses a sentence level stand-off based representation. As a result, the alignments are stored in a file separate from the source and target data files. Otherwise, this interface has almost the same functionality as the the word alignment interface. It allows words to be grouped together. It allows them (or the groups) to be assigned some tags. And it allows their alignments to be marked. It does differ, however, in the way these alignments are marked. Instead of drag-and-drop, it requires the user to select the contiguous text to be grouped as a unit and then to select a tag from a drop-down list. A table below allows the alignments to be marked. The selected units appear in this table. To make the process easier, the cells on the target side only list (as a drop-down list) the possible alignments, out of which the user has to select one. This interface has been used for creating a parallel corpus of tense, aspect and modality (TAM) markers (how they are translated) and also for marking translations of nominal compounds.

A tool associated with this interface makes it possible to gather some statistics. This tool can be used to find out, for example, what are the possible translations of a word group and how many times do they occur in the corpus. The statistics are about the source side, the target side and about the alignments (or translations). Such statistics were used for extracting features for training a CRF based model for automatically finding the translations of TAM markers (Singh et al., 2007).

This interface runs by default in what we call the *task mode*. In this mode, the user does not browse to input files directly. Instead, the user can create task groups and task lists within each group. Out of these, one task can be selected by the user to work on when the interface is started. The task configuration files are stored in the 'workspace' directory of Sanchay. The file listing the task groups is stored in 'workspace/parallel-corpus-markup' directory and is named 'task-groups.txt'. Within this are listed the files which, in turn, list the specific tasks. The description of each task includes the source and the target text files which are to be aligned as well as the lists of tags on the two sides. These tags are used to tag the aligned units. The input text files are plaintext files with one sentence per line and they are assumed to be sentence aligned with one-to-one mapping. These files are not changed during the annotation process. The output is in the form of three extra files. Two of them (the '.marked' files) contain the sentence level offsets and the tag indices of the units (words or word groups) that are marked for alignment. The third (the '.mapping' file) contains the actual alignments.

7. Syntactic Annotation Interface

It is not possible to describe all the features of this interface here as it is the interface with the most functionality in Sanchay. We briefly describe it here. We plan to prepare detailed manuals for this and other interfaces and post them on the Sanchay website.

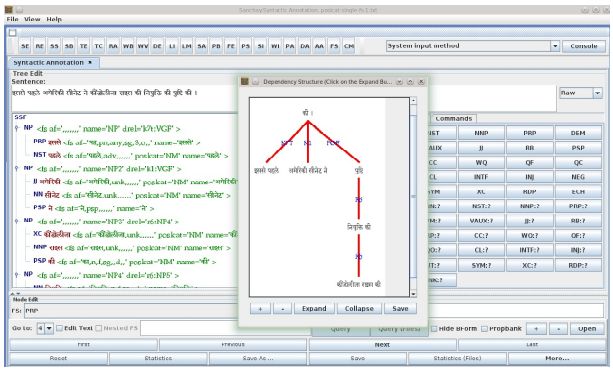


Figure 6. Syntactic Annotation Interface

This interface is meant for marking up the complete syntactico-semantic analysis of sentences. It can be used for annotating a sentence according to a phrase structure framework or a dependency structure framework. This includes information about the morphological analysis, POS tags, chunks, phrases, dependency relations and even certain kinds of semantic features. (There is also a separate but untested ProbBank annotation interface in the suite).

It is an easy to use interface that has evolved over the years based on the feedback of annotators and other kinds of users. The underlying representation is SSF. There is an API available for data in SSF and there are also various tools in the API to work on such data. There is a query language implementation in the API that allows easy search and modification of annotated data (Singh, 2012).

Figure 6 shows an example sentence in this interface, analyzed up to the dependency structure level. Dependency markup in this interface can be performed in a pop-up window by using the drag-and-drop mechanism, although there is a longer and more tedious method available too. The wealth of functionality in this interface makes it natural for us to use it for creating a parallel syntactic annotation interface.

8. Work In Progress: Parallel Syntactic Annotation Alignment Interface

This is an interface (Figure 7) that is still under construction. Only an initial prototype is ready. It uses the syntactic annotation interface on the source as well as the target side. The 'alignedTo' attribute here is used first at the document level and defines the scope of the alignments marked at the sentence and lower levels, and then at the setence and node levels. The input to this interface can be the output of either the sentence alignment interface or the word alignment interface.

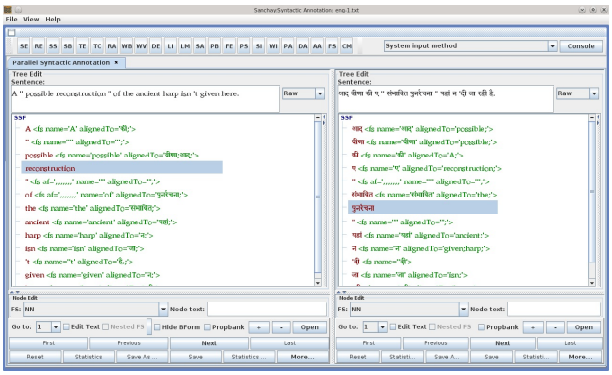


Figure 7. Parallel Syntactic Annotation Alignment Interface

The initial version of the interface simply has two panels, each containing the syntactic annotation interface, one for the source side and one for the target side. Alignments are marked again by simple drag-and-drop across the two panels. It can potentially be used to create parallel aligned treebanks such as the GALE Arabic-English Parallel Aligned Treebank[5].

The completion of this interface is the major future direction of this work. It will require resolving many issues that make it difficult to create such an interface. For example, how do we align dependency structures in this interface? Since the underlying representation is SSF, which has shallow parsed sentences at the core and has dependency relations marked by using attributes such as 'drel' (dependency relation), it is not directly possible to mark alignment of dependency nodes in the data representation (although it can be done in a visualizer). As the annotation of dependency relations is marked at the feature or attribute level, we essentially have to mark alignments also at the same level. This problem does not arise for phrase structure annotation, because such annotation is represented at the node level in SSF.

The above issue can be generalized to any information that is not directly represented at the node level. For example, alignments of parts of constituents even in the phrase structure framework poses the problem of not only representation, but also visualization in the interface. We do not yet have solutions for these issues. There may be other issues which we have not yet discovered and might find out once we try to complete the implementation of this interface.

The input to this interface is currently in the form of two files (source and target) in the Shakti Standard Format. The output is also stored in these same files. No additional files are created, but this may change as the interface develops.

[5] `http://catalog.ldc.upenn.edu/LDC2014T08`

9. Conclusion

Parallel aligned corpus can be a precious resource, especially if such corpus is annotated with linguistic analysis, as in a parallel aligned treebank. Even just sentence aligned parallel corpus is valuable enough to be the main resource needed for creating a statistical machine translation system. Therefore, user friendly annotation interfaces for creating parallel aligned corpora can be immensely useful. We described a set of annotation interfaces that ultimately aim to implement the entire process of the creation of parallel aligned treebanks, right from sentence alignment to word alignment to treebank alignment. This set of interfaces probably goes further than any other similar tool available in the open source domain. However, we find that, using our representation scheme (Shakti Standard Format or SSF), there are many issues which make it difficult to implement the entire treebank alignment process. One of them is that anything that is not represented directly at the node (document, sentence or word/chunk/constituent) level is difficult to mark up for alignment. Alignment of the dependency structure and non-node parts of phrase structure constituents are examples of this. The parallel markup interface, which uses the stand-off notation, does not have this problem, but it may be less user friendly.

Completing the implementation of the parallel syntactic annotation alignment interface is the main goal for the future. We also mentioned some other future directions such as completing the implementation of machine translation quality estimation measurement and post-editing functionality to the sentence alignment interface. To the word alignment interface, we can add the facility to tag alignment links as in the GALE Arabic-English Parallel Aligned Treebank. It can also be explored whether there is a better alternative to drag-and-drop.

Bibliography

Bharati, Akshar, Rajeev Sangal, Dipti Sharma, and Anil Kumar Singh. SSF: A Common Representation Scheme for Language Analysis for Language Technology Infrastructure Development. In *Proceedings of the COLING Workshop on Open Infrastructures and Analysis Frameworks for HLT (To Appear)*, 2014.

Brown, Peter F., Jennifer C. Lai, and Robert L. Mercer. Aligning Sentences in Parallel Corpora. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 169–176, 1991.

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

Corbí-Bellot, Antonio M., Mikel L. Forcada, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, Iñaki Alegria, Aingeru Mayor, and Kepa Sarasola. An Open-source Shallow-transfer Machine Translation Engine for the Romance Languages of Spain. In *Proceedings of the Tenth Conference of the European Association for Machine Translation*, pages 79–86, May 2005.

Gale, William A. and Kenneth Ward Church. A Propgram for Aligning Sentences in Bilingual Corpora. In *Proceedings of the 29th Annual Meeting of the Association of Computational Linguistics (ACL)*, 1991.

Gale, William A. and Kenneth W. Church. A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics*, 19(1):75–102, Mar. 1993. ISSN 0891-2017.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2007.

Li, Xuansong, Stephanie Strassel, Stephen Grimes, Safa Ismael, Mohamed Maamouri, Ann Bies, and Nianwen Xue. Parallel Aligned Treebanks at LDC: New Challenges Interfacing Existing Infrastructures. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, pages 1848–1855, 2012.

Moore, Robert C. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of the 5th conference of the Association for Machine Translation in the Americas (AMTA)*, pages 135–144, 2002.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

Singh, Anil Kumar. A Concise Query Language with Search and Transform Operations for Corpora with Multiple Levels of Annotation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, Instanbul, Turkey, 2012. ELRA.

Singh, Anil Kumar and Samar Husain. Comparison, Selection and Use of Sentence Alignment Algorithms for New Language Pairs. In *Proceedings of the ACL 2005 Workshop on Parallel Text*, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

Singh, Anil Kumar and Samar Husain. Exploring Translation Similarities for Building a Better Sentence Aligner. In *Proceedings of the 3rd Indian International Conference on Artificial Intelligence*, Pune, India, 2007.

Singh, Anil Kumar, Samar Husain, Harshit Surana, Jagadeesh Gorla, Chinnappa Guggilla, and Dipti Misra Sharma. Disambiguating Tense, Aspect and Modality Markers for Correcting Machine Translation Errors. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2007.

Tiedemann, Jörg. *Recycling Translations – Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing*. PhD thesis, Uppsala University, Uppsala, Sweden, 2003. Anna Sågvall Hein, Åke Viberg (eds): Studia Linguistica Upsaliensia.

Address for correspondence:
Anil Kumar Singh
nlprnd@gmail.com
Dept. of CSE, IIT (BHU)
Varanasi-221005, U.P., India

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 69–80

An open-source web-based tool for resource-agnostic interactive translation prediction

Daniel Torregrosa, Mikel L. Forcada, Juan Antonio Pérez-Ortiz

Departament de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, Spain

Abstract

We present a web-based open-source tool for interactive translation prediction (ITP) and describe its underlying architecture. ITP systems assist human translators by making context-based computer-generated suggestions as they type. Most of the ITP systems in literature are strongly coupled with a statistical machine translation system that is conveniently adapted to provide the suggestions. Our system, however, follows a *resource-agnostic* approach and suggestions are obtained from any unmodified *black-box* bilingual resource. This paper reviews our ITP method and describes the architecture of Forecat, a web tool, partly based on the recent technology of *web components*, that eases the use of our ITP approach in any web application requiring this kind of translation assistance. We also evaluate the performance of our method when using an unmodified Moses-based statistical machine translation system as the bilingual resource.

1. Introduction

Translation technologies are being increasingly used to assist human translators. Within this context, the objective of interactive translation prediction (ITP) tools (Foster et al., 1997; Barrachina et al., 2009) is to assist human translators in the translation of texts for dissemination by making context-based computer-generated suggestions as they type. Most works in the field of ITP have solely used specifically-adapted statistical machine translation (SMT) systems to obtain the suggestions. On the contrary, the *resource-agnostic* approach considered for the tool described in this paper explores how non-adapted *black-box* bilingual resources of any kind (a machine translation sys-

 Corresponding author: japerez@dlsi.ua.es
Cite as: Daniel Torregrosa, Mikel L. Forcada, Juan Antonio Pérez-Ortiz. An open-source web-based tool for resource-agnostic interactive translation prediction. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 69–80. doi: 10.2478/pralin-2014-0015.

tem, a translation memory, a bilingual dictionary, etc.) can be accommodated into an interoperable ITP framework.

This paper reviews the main aspects of our method and describes the architecture of Forecat, an HTML5 web tool, based on the recent technology of *web components* (see section 5.2), that eases the use of our ITP approach in any web application requiring this kind of translation assistance. To our knowledge, this is the first ITP tool that has been programmed as a web component.

The remainder of the paper is organised as follows. After reviewing the state-of-the-art in ITP in Section 2, we outline our resouce-agnostic proposal in Section 3. We then present unpublished results for a fully automatic evaluation of our approach in Section 4. After that, the architecture of an open-source web tool that implements our method is discussed in Section 5. Finally, we draw some conclusions in Section 6.

2. Related work

The systems which have most significantly contributed to the field of ITP are those built in the pioneering TransType project (Foster et al., 1997; Langlais et al., 2000), and its continuation, the TransType2 project (Macklovitch, 2006; Barrachina et al., 2009). These systems observe the current partial translation already typed by the user and, by exploiting an embedded SMT engine whose behaviour is modified to meet the needs of the ITP system, propose one or more continuations for the next words (or even the complete remainder of the sentence) that are compatible with the current sentence prefix. An automatic best-scenario evaluation (Barrachina et al., 2009) showed that it might theoretically be possible to use only 20–25% of the keystrokes needed in unassisted translation for English–Spanish translation (both directions) and around 45% for English–French and English–German. The results of user trials (Macklovitch, 2006) showed gains in productivity (measured in number of words translated per hour) of around 15–20%. A number of projects (Koehn, 2009; Ortiz-Martínez, 2011; Alabau et al., 2010, 2013) have recently continued the research where TransType2 left it off. As regards tools, both Caitra (Koehn, 2009) and the CASMACAT Workbench (Alabau et al., 2013) are web applications with an underlying SMT-based ITP system. Caitra consults the internal translation table of the SMT system Moses (Koehn et al., 2007) in order to show the most likely translation options to the human translator. Users can freely type the translation or accept any of the context-based suggestions generated *on the fly*: the most likely completion is directly shown next to the input field; the other translation candidates are also included in the interface so that users may click on any of them in order to incorporate it to the translation. The CASMACAT Workbench

also makes use of inner elements of Moses to offer assistance in the form of ITP, interactive editing with confidence information and adaptive translation models.[1]

Finally, many commercial translation memory systems also include the possibility of using ITP (see, for example, MT AutoSuggest,[2] a plug-in for the SDL Trados Studio tool).

3. A resource-agnostic interactive translation prediction approach

We propose a black-box treatment of the bilingual resources in contrast to the glass-box approaches found in literature. Unlike in the latter, access to the inner details of the translation system is consequently not necessary; this resource-agnostic approach minimises the coupling between the ITP tool and the underlying system and provides the opportunity to incorporate additional sources of bilingual information beyond purposely-designed SMT systems. These resources may include bilingual resources that cannot be adapted to produce a continuation for the remainder of the target-language sentence given a sentence prefix (for instance, because their code cannot be accessed or because they do not provide full sentence translations), but are able to supply the translation of a particular source-language segment. The underlying idea behind our approach is that resources such as machine translation (MT) cannot usually deliver appropriate translations at the sentence level, but their proposals usually contain acceptable segments that do not cover the whole sentence but which can be accepted by the user to assemble a good translation, saving as a result keystrokes, mouse actions or gestures, and, possibly, time. Note that by using the bilingual resources as black boxes that just provide translations, our system could be deprived of additional features that could prove useful if access to the inner details of the resource was possible.

A complete description of the method can be found in the paper by Pérez-Ortiz et al. (2014). What follows is an overview of its most relevant aspects.

Potential suggestions. Our method starts by splitting the source-language sentence S up into all the (possibly overlapping) segments of length $\mathsf{l} \in [1, \mathsf{L}]$, where L is the maximum source segment length measured in words.[3] The resulting segments are then translated by means of one or more bilingual resources. The set of *potential*

[1] Our tool is currently in an early stage of development, but some of these features could be incorporated into it in later stages. Note, however, that one of the major premises behind its development is to keep it as simple as possible so that it can been easily deployed as a standalone web component.

[2] http://www.codingbreeze.com/products/mtautosuggest/autosuggest_overview.htm

[3] Suitable values for L will depend on the bilingual resource: on the one hand, we expect higher values of L to be useful for high-quality MT systems, such as those translating between closely related languages, since adequate translations may stretch to a relatively large number of words; on the other hand, L should be kept small for resources such as dictionaries or low-quality MT systems whose translations quickly deteriorate as the length of the input segment increases.

proposals P^{S} for sentence S is made up of pairs comprising the translation of each segment and the position in the input sentence of the first word of the corresponding source-language segment. For example, the source-language segments obtained when translating the English sentence S = "My tailor is healthy" into Spanish with $\mathsf{L} = 3$ are *My*, *My tailor*, *My tailor is*, *tailor*, *tailor is*, *tailor is healthy*, *is*, *is healthy*, and *healthy*; the corresponding set of potential suggestions P^{S} is made up of (*Mi*, 1), (*Mi sastre*, 1), (*Mi sastre es*, 1), (*sastre*, 2), (*sastre es*, 2), (*sastre está sano*, 2), (*es*, 3), (*está sano*, 3), and (*sano*, 4). We shall represent the $\mathfrak{i}$-th suggestion as $\mathsf{p}_\mathfrak{i}$, its target-language segment as $\mathsf{t}(\mathsf{p}_\mathfrak{i})$ and its corresponding source-language word position as $\sigma(\mathsf{p}_\mathfrak{i})$.

Compatible suggestions. Let $\mathsf{P}^{\mathsf{S}}_{\mathsf{C}}(\hat{w})$ be the subset of P^{S} including the *compatible suggestions* which can be offered to the user after typing $\hat{w}$ as the prefix of the current word in the translated sentence T. The elements of $\mathsf{P}^{\mathsf{S}}_{\mathsf{C}}(\hat{w})$ are determined by considering only those suggestions in P^{S} that have the already-typed word prefix as their own prefix:

$$\mathsf{P}^{\mathsf{S}}_{\mathsf{C}}(\hat{w}) = \{\mathsf{p}_\mathfrak{i} \in \mathsf{P}^{\mathsf{S}} : \hat{w} \in \mathrm{Prefix}(\mathsf{t}(\mathsf{p}_\mathfrak{i}))\}$$

For example, in the case of the translation of the previous English sentence, if the user types an *M*, the set of compatible suggestions $\mathsf{P}^{\mathsf{S}}_{\mathsf{C}}(M, 1)$ will contain the suggestions with target-language segments *Mi*, *Mi sastre* and *Mi sastre es*, since they are the only proposals in P^{S} starting with an *M*.

Except for very short sentences, the number of compatible suggestions usually exceeds what users are expected to tolerate. Therefore, adequate strategies are necessary in order to reduce the number of suggestions eventually offered to the user to an appropriate value. The degree of success that can be achieved in this task will be explored in greater depth in future work, but a naïve *distance-based* approach that ranks the suggestions in $\mathsf{P}^{\mathsf{S}}_{\mathsf{C}}(\hat{w})$ based solely on the position j of the current word in the target sentence has already provided interesting results (Pérez-Ortiz et al., 2014).

Ranking suggestions. Under the distance-based approach, suggestions $\mathsf{p}_\mathfrak{i}$ whose source position $\sigma(\mathsf{p}_\mathfrak{i})$ is closer (in terms of the absolute difference) to the position j in the target sentence of $\hat{w}$ are prioritised.[4] For example, in the case of the translation mentioned above, if the user has just typed *Mi s* and is introducing the second word of the translation, suggestions starting with *sastre* (originated at source position 2) will be ranked before those starting with *sano* (originated at position 4).

Let M be the maximum number of suggestions that will eventually be offered to the human translator. For the small values of M that are acceptable for a user-friendly

[4]We cannot in principle expect this ranker to work reasonably well on unrelated languages with very divergent grammatical structures (e.g., when translating a language with a verb–subject–object order into another one with a subject–verb–object order).

interface, it may easily happen that all the suggestions offered are obtained starting at the same source position (that closest to the current target position) although better suggestions from different positions exist. In order to mitigate the impact of this, the *distance-based* ranking is partially relaxed in the experiments in this paper in a similar way as proposed by Pérez-Ortiz et al. (2014): only the longest and the shortest suggestions from each position are in principle chosen; the rest of the suggestions, if any, would only be offered if the number of maximum offered suggestions M is not reached after exhausting the longest and shortest proposals from all compatible positions.

4. Experiments

Although the main purpose of this paper is to introduce the architecture of our ITP tool, in this section we show the results of an automatic evaluation carried on using a black-box phrase-based SMT system as the only bilingual resource. This evaluation will provide an idea of the best results attainable with our method by human translators. The approach followed for the automatic evaluation is identical to that described by Langlais et al. (2000), in which a parallel corpus with pairs of sentences was used. In the context of our automatic evaluation, each source-language sentence S is used as the input sentence to be translated and the corresponding target-language T is considered as the output a user is supposed to have in mind and stick to while *typing*. The longest suggestion in the list of offered suggestions which concatenated to the already typed text results in a new prefix of T is always used. If there are no suggestions at a particular point, then the automatic evaluation system *continues typing* according to T. The performance of our system is measured by using the *keystroke ratio* (KSR), that is, the ratio between the number of keystrokes and the length of the translated sentence (Langlais et al., 2000).[5]

We have evaluated whether the domain of the corpora used to train the Moses-based SMT system (Koehn et al., 2007) affects the KSR. For this, the phrase-based SMT systems have been trained in the standard way, more exactly as described by Haddow and Koehn (2012).[6] Only the best translation proposed by Moses for each segment has been considered in our experiments. Tests have been performed with 3 different systems: one trained and tuned with out-of-domain corpora, another trained with out-of-domain corpora but tuned with in-domain corpora, and a third one trained and tuned with in-domain corpora.

Both L and M are set to 4 following the recommendations from previous automatic evaluations (Pérez-Ortiz et al., 2014). On the one hand, $L = 4$ represents a good compromise between computational load and usefulness of the suggestions; only marginal

[5] A lower KSR represents a greater saving in keystrokes.

[6] The method for the compression of translation phrase tables proposed by Junczys-Dowmunt (2012) has also been used.

performance improvement is attained when incrementing it, with the drawback of having more equivalents to obtain through the bilingual resources, more suggestions to filter out, etc. On the other hand, the performance with $M = 4$ has proved to be close to that obtained considering all the possible candidate suggestions without posing a significant hindrance for the human translators.

Corpora. From all the pairs considered in the work by Haddow and Koehn (2012)[7], English–Spanish (en–es) and English–Czech (en–cs) were chosen (in both translation directions), as they are, respectively, the best and worst performing pairs in that work. For English–Czech and English–Spanish, we used the most up-to-date version of the corpora used by Haddow and Koehn (2012): the v7 release of the Europarl corpora.[8] and the ACL2013 News Commentary corpora[9]. 2 000 sentences from each corpora were selected as tuning set, and 15 000 from News Commentary were extracted as test. The rest of the corpora was used as training set: in the case of Europarl, 623 913 sentences for English–Czech, and 1 912 074 for English–Spanish; in the case of News Commentary, 122 720 sentences for English–Czech and 155 760 for English–Spanish.

Results. The KSR values for the automatic evaluation are shown in table 1. As expected, English–Spanish performed better (savings in keystrokes up to 48%) than English–Czech (savings in keystrokes up to 31%) because it is generally easier to translate between English and Spanish, and the available corpora were larger in this case as well. Though statistically significant, the differences between the different in-domain and out-of-domain systems are relatively small. For the purposes of comparison, the rule-based MT system Apertium (Forcada et al., 2011) has been reported (Pérez-Ortiz et al., 2014) to provide a KSR of 0.76 for English–Spanish and 0.70 for Spanish–English when using, in both cases, $L = 4$ and $M = 4$; note, however, that Apertium is an already-built *general-purpose* MT system, which makes it impossible to differentiate between results for in-domain or out-of-domain scenarios.

5. Technical Issues and Implementation

In this section we describe Forecat, an open-source web-based tool that we have implemented to demonstrate the validity of our ITP approach and incorporate its use in real-life applications. Forecat can be used in three different ways as described next.

Firstly, it can be used as a simple web application for computer-assisted translation. Under this perspective, our tool has a web interface similar to that in the projects

[7] In their paper, the influence of in-domain and out-of-domain corpora when training SMT systems was evaluated.

[8] http://www.statmt.org/europarl/

[9] http://www.statmt.org/wmt13/translation-task.html

	en–es	es–en	en–cs	cs–en
ep	0.70	0.53	0.78	0.69
ep+nc	0.62	0.52	0.75	0.66
nc	0.62	0.52	0.76	0.64

Table 1. KSR values from the automatic evaluation of our ITP method. In the table, ep *stands for the system trained and tuned with Europarl (out-of-domain);* nc *for the one trained and tuned with News Commentary (in-domain); and* ep+nc *for the one trained with Europarl but tuned with News Commentary. In all cases, the test set consisted of sentences extracted from the News Commentary corpus. For the purposes of comparison, the rule-based MT system Apertium (Forcada et al., 2011) has been reported (Pérez-Ortiz et al., 2014) to provide a KSR of 0.76 for English–Spanish and 0.70 for Spanish–English.*

discussed in Section 2: users freely type the translation of the source sentence, and are offered suggestions *on the fly* in a drop-down list with items based on the current prefix; users may accept these suggestions (using cursor keys, the mouse or specific hot keys) or ignore them and continue typing. A screenshot of the interface is shown in Figure 1. Despite the cognitive load inherent to any predictive interface, the interface is easy and intuitive to use, even for inexperienced users, as can be deduced from the results of a preliminary user trial (Pérez-Ortiz et al., 2014).

Secondly, it can be deployed as a set of web services with an *application programming interface* (API) that provides the basic functionalities for integrating our ITP method in third-party applications. The web API that can be deployed with Forecat has four GET services that use JSON-formatted[10] data. First, the list of available language pairs has to be obtained by the client. Then, the sentence to translate is submitted to the server and the total number of resulting proposals is returned. After that, the list of suggestions to be offered according to the current typed prefix is requested. If the user selects one of the suggestions, the server has to be notified about this by the client through a fourth service.

Finally, we have recently started to build a *web component* from the existing code of Forecat. Web components comply with a number of standards[11] whose objective is to enable fully encapsulated and reusable components for the web.[12] See section 5.2 for additional details about Forecat's web component.

[10] http://www.ecma-international.org/publications/standards/Ecma-404.htm

[11] See http://www.w3.org/TR/components-intro/ for more information.

[12] Web components are called upon to dramatically change how developers build web applications by allowing them to declaratively incorporate independent widgets into their applications with a number of possibilities and advantages not possible with today's established technologies.

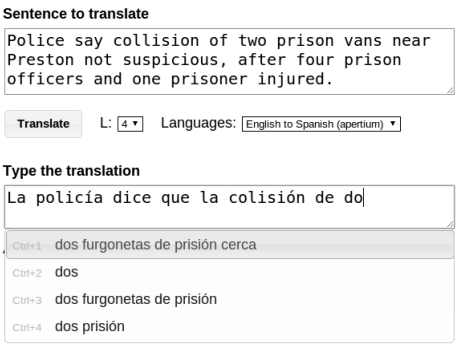


Figure 1. Screenshot of the interface of our ITP web application showing a translation in progress with some suggestions being offered. The top text box contains the source sentence, whereas users type the translation into the bottom box.

5.1. Programming Languages and Frameworks

Forecat's logic is mostly written in Java with the help of the Google Web Toolkit[13] (GWT), an open-source framework for developing web applications. At the core of the framework is a compiler which translates Java code to JavaScript code which runs flawlessly in current browsers. This allows for a twofold use of the Java code that implements our resource-agnostic ITP method: on the one hand, it can be used locally in Java when performing the automatic evaluation of our approach or when a client calls the corresponding web services; on the other hand, the same code (except for the module responsible of the translation of the segments) can be executed on the browser in JavaScript when human translators interact with the tool either through the web aplication or the web component, thus improving the performance of the tool.

A small part of Forecat, the one dealing with the interface of both the web application or the web component, has been originally written in JavaScript, since we decided not to use GWT for programming the elements of the interface in order to decouple them from the implementation of the method.

5.2. Web Component

We envision Forecat's future as a web component more than as a full web application. This very recent technology allows us to define an encapsulated and interoperable HTML5 custom element (in this case, a translation box) which can easily be imported into any webpage to provide the functionalities of our ITP approach. Forecat code

[13] http://www.gwtproject.org/

already includes a working first prototype of an ITP component.[14] For this, it uses Polymer,[15] an open-source Javascript library that simplifies the creation of web components and makes it possible to benefit from them even in those browsers which do not implement the latest version of the underlying standards. The interface of the web component is similar to the bottom box in Figure 1, but the tool works in this case as a standalone widget. The following code shows an example of a simple webpage including a translation box that will offer suggestions as the user translates the sentence *My tailor is healthy* into Spanish, if this pair is available in the component.

```
<!DOCTYPE html>
<html>
<head>
  <script src="bower_components/platform/platform.js"></script>
  <link rel="import" href="elements/translation-box.html">
</head>
<body>
<translation-box id="itp"></translation-box>
<script>
var component = document.querySelector('#itp');
component.addEventListener('languagesReady', function (e) {
  if (contains(e.detail,"en-es")) {
    component.pair= "en-es";
    component.sourceText= "My tailor is healthy.";
  }
});
</script>
</body>
```

The `script` element loads the Polymer library. The next line *imports* our web component which provides the custom element `translation-box` used in the document body. The JavaScript code waits for the `languagesReady` event fired by the component when it is ready to operate and then changes its public attributes `pair` and `sourceText`. The component observes changes in these attributes and then uses their values to obtain the suggestions that will be offered to the translator. The declaration of the component includes a *template* that contains an editable `div` element where the translation will be typed.

[14] See the libjs/component directory in the Forecat code.

[15] http://www.polymer-project.org/

5.3. License Choice and Download

Forecat is licensed under version 3 of the GNU Affero General Public License[16] (AGPL). This license is fully compatible with the GNU General Public License (GPL) and equally proposed by the Free Software Foundation, which in fact recommends[17] that "developers consider using the GNU AGPL for any software which will commonly be run over a network". AGPL has been suggested as a means to close a *loophole* in the ordinary GPL which does not force organisations to distribute derivative code when it is only deployed as a web service. The entire code of the application can be downloaded from the Github repository.[18]

6. Conclusions and future work

Resource-agnostic interactive translation prediction (ITP) is a low-cost approach for computer-assisted translation. Forecat, the open-source resource-agnostic ITP tool whose architecture has been discussed in this paper provides a convenient implementation of the ideas behind this approach in the form of a web application, a number of web services, and a web component that can be easily integrated into third-party solutions.

We plan to improve the ranking strategy shown in Section 3 by automatically detecting the part of the input sentence being translated at each moment so that segments that originate in those positions are prioritised. We intend to achieve this by combining word alignment and distortion models. On the one hand, the former will be used to determine the alignments between the last words introduced by the user and the words in the input sentence. On-the-fly, light alignment models have been proposed (Esplà-Gomis et al., 2012) which do not require parallel corpora and are based on the translation of all the possible segments of the sentence with the help of black-box bilingual resources; these models would fit nicely into our ITP method. On the other hand, distortion models, as those proposed by Al-Onaizan and Papineni (2006), will be used to predict which source words will be translated next, partly by using information from the alignment model.

We also plan to explore the impact of simultaneously using different black-box bilingual resources. Different strategies will be evaluated in order to integrate the available resources: combining the findings of the various translation resources into a single suggestion as done by Nirenburg and Frederking (1994) in their multi-engine MT system; using confidence-based measures in order to select the most promising translations as performed by Blatz et al. (2004); or predicting the best candidates for the translation of each particular segment by using only source-language information,

[16] http://www.gnu.org/licenses/agpl-3.0.html

[17] http://www.fsf.org/licensing/licenses/

[18] https://github.com/jaspock/forecat/

thus avoiding the need to consult every available resource, as explored by Sánchez-Martınez (2011).

Although there is room for many future improvements, a distance-based ranker, in spite of its simplicity, already provides encouraging results: according to the best results of our automatic experiments, when a maximum of $M = 4$ suggestions are offered and the system selects the longest one that matches the reference translation, around 25–50% keystrokes could be saved depending on the language pair and on the domain of the corpora used to train the SMT system used as bilingual resource.

Acknowledgments

This work has been partly funded by the Spanish Ministerio de Economía y Competitividad through project TIN2012-32615.

Bibliography

Al-Onaizan, Yaser and Kishore Papineni. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 529–536. Association for Computational Linguistics, 2006.

Alabau, Vicent, Daniel Ortiz-Martínez, Alberto Sanchis, and Francisco Casacuberta. Multimodal interactive machine translation. In *ICMI-MLMI '10: Proceedings of the 2010 International Conference on Multimodal Interfaces*, 2010.

Alabau, Vicent, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González-Rubio, Philipp Koehn, Luis A. Leiva, Bartolomé Mesa-Lao, Daniel Ortiz, Herve Saint-Amand, Germán Sanchis-Trilles, and Chara Tsoukala. CASMACAT: An open source workbench for advanced computer aided translation. *Prague Bull. Math. Linguistics*, 100:101–112, 2013.

Barrachina, Sergio, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28, 2009.

Blatz, John, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220401. URL http://dx.doi.org/10.3115/1220355.1220401.

Esplà-Gomis, Miquel, Felipe Sánchez-Martínez, and Mikel L. Forcada. Using external sources of bilingual information for on-the-fly word alignment. Technical report, Departament de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, 2012.

Forcada, Mikel L, 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. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144, 2011.

Foster, George F., Pierre Isabelle, and Pierre Plamondon. Target-text mediated interactive machine translation. *Machine Translation*, 12(1-2):175–194, 1997.

Haddow, Barry and Philipp Koehn. Analysing the effect of out-of-domain data on SMT systems. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, Montreal, Canada, June 2012. Association for Computational Linguistics.

Junczys-Dowmunt, Marcin. Phrasal rank-encoding: Exploiting phrase redundancy and translational relations for phrase table compression. *The Prague Bulletin of Mathematical Linguistics*, 98:63–74, 2012.

Koehn, Philipp. A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 17–20, 2009.

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

Langlais, Philippe, Sébastien Sauvé, George Foster, Elliott Macklovitch, and Guy Lapalme. Evaluation of TransType, a computer-aided translation typing system: a comparison of a theoretical-and a user-oriented evaluation procedures. In *Conference on Language Resources and Evaluation (LREC)*, 2000.

Macklovitch, Elliott. TransType2: The last word. In *Proceedings of the 5th International Conference on Languages Resources and Evaluation (LREC 06)*, pages 167–172, 2006.

Nirenburg, Sergei and Robert Frederking. Toward multi-engine machine translation. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 147–151, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075842. URL http://dx.doi.org/10.3115/1075812.1075842.

Ortiz-Martínez, Daniel. *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. PhD thesis, Universitat Politècnica de València, 2011.

Pérez-Ortiz, Juan Antonio, Daniel Torregrosa, and Mikel Forcada. Black-box integration of heterogeneous bilingual resources into an interactive translation system. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-Assisted Translation*, 2014.

Sánchez-Martınez, Felipe. Choosing the best machine translation system to translate a sentence by using only source-language information. In *Proceedings of the 15th Annual Conference of the European Associtation for Machine Translation*, pages 97–104, 2011.

Address for correspondence:
Juan Antonio Pérez-Ortiz
japerez@dlsi.ua.es
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
Carretera Sant Vicent del Raspeig s/n
03690 Sant Vicent del Raspeig, Alacant, Spain

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 81–92

OxLM: A Neural Language Modelling Framework for Machine Translation

Paul Baltescu[a], Phil Blunsom[a], Hieu Hoang[b]

[a] University of Oxford, Department of Computer Science
[b] University of Edinburgh, School of Informatics

Abstract

This paper presents an open source implementation[1] of a neural language model for machine translation. Neural language models deal with the problem of data sparsity by learning distributed representations for words in a continuous vector space. The language modelling probabilities are estimated by projecting a word's context in the same space as the word representations and by assigning probabilities proportional to the distance between the words and the context's projection. Neural language models are notoriously slow to train and test. Our framework is designed with scalability in mind and provides two optional techniques for reducing the computational cost: the so-called class decomposition trick and a training algorithm based on noise contrastive estimation. Our models may be extended to incorporate direct n-gram features to learn weights for every n-gram in the training data. Our framework comes with wrappers for the `cdec` and `Moses` translation toolkits, allowing our language models to be incorporated as *normalized* features in their decoders (inside the beam search).

1. Introduction

Language models are statistical models used to score how likely a sequence of words is to occur in a certain language. They are central to a number of natural language applications, including machine translation. The goal of a language model in a translation system is to ensure the fluency of the output sentences.

[1] Our code is publicly accessible at: `https://github.com/pauldb89/oxlm`

 Corresponding author: `paul.baltescu@cs.ox.ac.uk`
Cite as: Paul Baltescu, Phil Blunsom, Hieu Hoang. OxLM: A Neural Language Modelling Framework for Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 81–92.
doi: 10.2478/pralin-2014-0016.

Most machine translation systems today use highly efficient implementations of n-gram language models (Heafield, 2011; Stolcke, 2002). N-gram language models represent the target vocabulary as a discrete set of tokens and estimate the conditional probabilities $P(w_i|w_{i-1},\ldots,w_{i-n})$ via frequency counting. Kneser-Ney smoothing (Chen and Goodman, 1999) is typically used to ameliorate the effect of data sparsity. Querying n-gram language models is extremely fast as the only operations involved are hashtable or trie lookups, depending on the implementation.

Neural language models (Bengio et al., 2003) are a more recent class of language models which use neural networks to learn distributed representations for words. Neural language models project words and contexts into a continuous vector space. The conditional probabilities $P(w_i|w_{i-1},\ldots,w_{i-n})$ are defined to be proportional to the distance between the continuous representation of the word w_i and the context $w_{i-1},\ldots,w_{i-n}$. Neural language models learn to cluster word vectors according to their syntactic and semantic role. The strength of neural language models lies in their ability to generalize to unseen n-grams, because similar words will share the probability of following a context. Neural language models have been shown to outperform n-gram language models using intrinsic evaluation (Chelba et al., 2013; Mikolov et al., 2011a; Schwenk, 2007) or as part of other natural language systems such as speech recognizers (Mikolov et al., 2011a; Schwenk, 2007). In machine translation, it has been shown that neural language models improve translation quality if incorporated as an additional feature into a machine translation decoder (Botha and Blunsom, 2014; Vaswani et al., 2013) or if used for n-best list rescoring (Schwenk, 2010). Querying a neural language model involves an expensive normalization step linear in the size of the vocabulary and scaling this operation requires special attention in order for a translation system to maintain an acceptable decoding speed.

The goal of this paper is to introduce an open source implementation of a feed forward neural language model. As part of our implementation, we release wrappers which enable the integration of our models as *normalized* features in the `cdec` (Dyer et al., 2010) and `Moses` (Koehn et al., 2007) decoders. Our framework is designed with scalability in mind and provides two techniques for speeding up training: class-based factorization (Morin and Bengio, 2005) and noise contrastive estimation (Mnih and Teh, 2012). The class decomposition trick is also helpful for reducing the cost of querying the language model and allows a decoder incorporating our feature to maintain an acceptable decoding speed. In addition to this, our framework optionally extends neural language models by incorporating direct n-gram features (similar to Mikolov et al. (2011a)).

2. Related work

In this section, we briefly analyze three open source neural language modelling toolkits. We discuss how each implementation is different from our own and show where our approach has additional strengths.

CSLM (Schwenk, 2010) is an open source toolkit implementing a continuous space language model which has a similar architecture to our own. CSLM employs a *short-list* to reduce the computational cost of the normalization step. A short-list contains the most frequent words in the training corpus. Schwenk (2010) reports setting the size of the short-list to 8192 or 12288 words. The continuous space language model is used to predict only the words in the short-list, while the remaining words are scored using a back-off n-gram language model. We believe this optimization hurts the model where the potential benefit is the greatest, as the strength of neural language models relies in predicting rare words. In addition to this, CSLM may only be used to rescore n-best lists and cannot be incorporated as a feature in a decoder.

NPLM (Vaswani et al., 2013) is another open source implementation of a neural language model. In contrast to our implementation, Vaswani et al. (2013) do not explicitly normalize the values produced by their model and claim that these scores can be roughly interpreted as probabilities. In practice, we observed that unnormalized scores do not sum up to values close to 1 when the predicted word is marginalized over the vocabulary. Our approach trades decoding speed for the guarantee of using properly scaled feature values.

RNNLM (Mikolov et al., 2011b) is an open source implementation of a recurrent neural language model. Recurrent neural language models have a somewhat different architecture where the hidden layer at step i is provided as input to the network at step $i + 1$. RNNLM uses the class decomposition trick to speed up queries. The toolkit also allows extending the language models with direct n-gram features. RNNLM has been successfully used in speech recognition tasks (Mikolov et al., 2011a), and Auli and Gao (2014) show that recurrent neural language models considerably improve translation quality when integrated as an unnormalized feature into a decoder.

3. Model description

Our implementation follows the basic architecture of a log-bilinear language model (Mnih and Hinton, 2007). We define two vector representations $\mathbf{q}_w, \mathbf{r}_w \in \mathbb{R}^D$ for every word w in the vocabulary V. $\mathbf{q}_w$ represents w's syntactic and semantic role when the word is part of the conditioning context, while $\mathbf{r}_w$ is used to represent w's role as a prediction. For some word w_i in a given corpus, let h_i denote the conditioning context $w_{i-1}, \ldots, w_{i-n}$. To find the conditional probability $P(w_i|h_i)$, our model first computes a context projection vector:

$$\mathbf{p} = \sum_{j=1}^{n-1} C_j \mathbf{q}_{h_{ij}}, \tag{1}$$

where $C_j \in \mathbb{R}^{D \times D}$ are position-specific transformation matrices. Our implementation provides an optional flag which applies a component-wise sigmoid non-linearity to the projection layer, transforming the model into one similar to Bengio et al. (2003).

The model computes a set of similarity scores indicating how well each word $w \in V$ matches the context projection of h_i. The similarity score is defined as:

$$\phi(w, h_i) = \mathbf{r}_w^T \mathbf{p} + b_w, \tag{2}$$

where b_w is a bias term incorporating the prior probability of w. The similarity scores are transformed into a probability distribution using the *softmax* function:

$$P(w_i|h_i) = \frac{\exp(\phi(w_i, h_i))}{\sum_{w \in V} \exp(\phi(w, h_i))} \tag{3}$$

The complete set of parameters is $(C_j, Q, R, \mathbf{b})$, where $Q, R \in \mathbb{R}^{D \times |V|}$ and $\mathbf{b} \in \mathbb{R}^{|V|}$. The model is trained using minibatch stochastic gradient descent to minimize the negative log-likelihood of the training data. L_2 regularization is used to prevent overfitting.

3.1. Class based factorization

The difficulty of scaling neural language models lies in optimizing the normalization step illustrated in Equation 3. Our implementation relies on class based decomposition (Morin and Bengio, 2005; Goodman, 2001) to reduce the cost of normalization. We partition our vocabulary in K classes $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ using Brown clustering (Liang, 2005; Brown et al., 1992) such that $V = \bigcup_{i=1}^{K} \mathcal{C}_i$ and $\mathcal{C}_i \cap \mathcal{C}_j = \varnothing, \forall 1 \leq i < j \leq K$. We define the conditional probability as:

$$P(w_i|h_i) = P(c_i|h_i) P(w_i|c_i, h_i), \tag{4}$$

where c_i is the index of the class w_i is assigned to, i.e. $w_i \in \mathcal{C}_{c_i}$. We associate a vector representation $\mathbf{s}_c$ and a bias term t_c for each class c. The class conditional probability is computed reusing the prediction vector $\mathbf{p}$ by means of a scoring function $\psi(c, h_i) = \mathbf{s}_c^T p + t_c$. Each conditional distribution is now normalized separately:

$$P(c_i|h_i) = \frac{\exp(\psi(c_i, h_i))}{\sum_{j=1}^{K} \exp(\psi(c_j, h_i))} \tag{5}$$

$$P(w_i|c_i, h_i) = \frac{\exp(\phi(w_i, h_i))}{\sum_{w \in C_{c_i}} \exp(\phi(w, h_i))} \tag{6}$$

The best performance is achieved when $K \approx \sqrt{|V|}$ and the word classes have roughly equal sizes. In that case, the normalization cost for predicting a word is reduced from $O(|V|)$ to $O(\sqrt{|V|})$.

3.2. Noise contrastive estimation

Training neural language models using stochastic gradient descent is slow because the entire matrix $R \in \mathbb{R}^{D \times |V|}$ is modified with every gradient update. The class based factorization reduces the cost of computing the gradient of R to $O(D \times \sqrt{|V|})$. In our implementation, we provide an optimization for computing the gradient updates based on noise contrastive estimation, a technique which does not involve normalized probabilities (Mnih and Teh, 2012). Noise contrastive training can be used with or without class based decomposition.

The key idea behind noise contrastive estimation is to reduce a density estimation problem to a classification problem, by training a binary classifier to discriminate between samples from the data distribution and samples from a known noise distribution. In our implementation, we draw the noise samples n_i from the unigram distribution denoted by $P_n(w)$. Following Mnih and Teh (2012), we use k times more noise samples than data samples, where k is specified via an input argument. The posterior probability that a word is generated from the data distribution given its context is:

$$P(C = 1|w_i, h_i) = \frac{P(w_i|h_i)}{P(w_i|h_i) + kP_n(w_i)} \tag{7}$$

Mnih and Teh (2012) show that the gradient of the classification objective:

$$J(\theta) = \sum_{i=1}^{m} \log p(C = 1|\theta, w_i, h_i) + \sum_{i=1}^{km} \log p(C = 0|\theta, n_i, h_i) \tag{8}$$

is an approximation which converges to the maximum likelihood gradient as $k \to \infty$. Noise contrastive estimation allows us to replace the normalization terms with model parameters. Mnih and Teh (2012) showed that setting these parameters to 1 results in no perplexity loss. In our implementation of noise contrastive training, we simply ignore the normalization terms, but this optimization is not applicable at test time.

3.3. Direct n-gram features

Direct features (or connections) for unigrams were originally introduced in neural language models by Bengio et al. (2003). Mikolov et al. (2011a) extend these features to n-grams and show they are useful for reducing perplexity and improving word error rate in speech recognizers. Direct n-gram features are reminiscent of maximum entropy language models (Berger et al., 1996) and are sometimes called maximum entropy features (e.g. in Mikolov et al. (2011a)).

The basic idea behind direct features is to define a set of binary feature functions $\mathbf{f}(w, h)$ and to assign each function a weight from a real valued vector $\mathbf{u}$. In our implementation, we define a feature function $f(w, h)$ for every n-gram in the training

data, up to some order specified by an input argument. To account for word classes, we also define a set of n-gram features $\mathbf{g}_c(w, h)$ and a vector of weights $\mathbf{v}_c$ for each word cluster c. An n-gram (w, h) has a corresponding feature function $g_c(w, h)$ only if $w \in \mathcal{C}_c$. To incorporate the features into our model, we update the scoring functions as follows:

$$\psi(c_i, h_i) = \mathbf{s}_{c_i}^\top \mathbf{p} + t_{c_i} + \mathbf{u}^\top \mathbf{f}(w_i, h_i) \tag{9}$$

$$\phi(w_i, h_i) = \mathbf{r}_{w_i}^\top \mathbf{p} + b_{w_i} + \mathbf{v}_{c_i}^\top \mathbf{g}_{c_i}(w_i, h_i) \tag{10}$$

Otherwise, our model definition remains unchanged. The weight vectors $\mathbf{u}$ and $\mathbf{v}_c$ are learned together with the rest of the parameters using gradient descent. From the perspective of the machine translation system, the language model is extended to learn weights for every n-gram in the training data, weights which bear a similar role to the frequency counts used by traditional n-gram language models.

4. Implementation details

4.1. Training language models

Our language modelling framework is implemented in `C++`. Compiling it will result in a number of binaries. `train_sgd`, `train_factored_sgd` and `train_maxent_sgd` are used for training language models, while the other binaries are useful for evaluation and debugging. Due to lack of space, we will only discuss the most important arguments provided in the training scripts. For a complete list of available options and a short description of each, any binary may be run with the `--help` argument. Examples of intended usage and recommended configurations are released together with our code.

`train_sgd` is used to train neural language models without class factorization or direct features. The binary reads the training data from the file specified via the `--input` parameter. The optional `--test-set` parameter is used to specify the file containing the test corpus. If specified, the training script computes the test set perplexity every 1000 minibatches and at the end of every training epoch. The `--model-out` argument specifies the path where the language model is saved. The language model is written to disk every time the test set perplexity reaches a new minimum. The `--order` parameter specifies the order of the model, the `--word-width` parameter specifies the size of the distributed representations and the `--lambda-lbl` parameter is the inverse of the variance for the L_2 regularizer. If `--noise-samples` is set to 0, the model is trained using stochastic gradient descent. Otherwise, the parameter specifies the number of noise samples drawn from the unigram distribution for each training instance during noise contrastive training.

Factored models are trained with the `train_factored_sgd` binary. In addition to the previous arguments, this script includes the `--class-file` option which points

to the files containing the Brown clusters. The expected format matches the output format of Liang (2005)'s agglomerative clustering tool[2]. If the `--class-file` argument is not specified, the user is required to set the `--classes` argument. In this case, the word clusters are obtained using frequency binning.

Factored models incorporating direct features are trained with `train_maxent_sgd`. We implement two types of feature stores for storing the weights of the n-gram features. Sparse feature stores use identity mapping to map every n-gram with its corresponding weight. Collision stores hash the n-grams to a lower dimensional space instead, leading to potential collisions. If configured correctly using the `--hash-space` parameter, collision stores require less memory than sparse feature stores, without any perplexity loss. If the argument is set to 0, sparse stores are used instead. The `--min-ngram-freq` argument may be used to ignore n-grams below a frequency threshold, while `--max-ngrams` may be used to restrict the number of direct features to the most frequent n-grams in the training data.

4.2. Feature wrappers for `cdec` and `Moses`

Our language models may be incorporated into the `cdec` and `Moses` decoders as normalized features to score partial translation hypotheses during beam search. The decoders often keep the conditioning context unchanged and create new translation hypotheses by adding new words at the end of the conditioning context. We significantly speed up decoding by caching the normalization terms to avoid recomputing them every time a new word is added after the same context. The normalization cache is reset every time the decoders receive a new sentence as input.

Compiling our framework results in the `libcdec_ff_lbl.so` shared library which is used to dynamically load our language models as a feature in the `cdec` decoder. To load the feature, a single line must be added to the decoder configuration file specifying the path to the shared library, the file containing the language model and the type of the language model (standard, factored or factored with direct features). A complete `cdec` integration example is provided in the documentation released with our code.

The feature wrapper for `Moses` is included in the `Moses` repository[3]. To include our language models in the decoder, `Moses` must be compiled with the `--with-lbllm` argument pointing to the location of the `oxlm` repository. The decoder configuration file must be updated to include the feature definition, the path to the file containing the language model and the initial feature weight. A complete example on how to integrate our language models in `Moses` is provided in the documentation released with our code.

---

[2]The tool is publicly available at: `https://github.com/percyliang/brown-cluster`

[3]The feature wrapper is accessible here: `https://github.com/moses-smt/mosesdecoder/tree/master/moses/LM/oxlm`

4.3. Optimizations

Our framework includes several smaller optimizations designed to speed up training and testing. We provide an optional `--diagonal-contexts` argument which informs the framework to learn a model with diagonal context matrices. This optimization significantly speeds up querying the language model and helps the training algorithm converge after fewer iterations without any loss in perplexity.

Our implementation leverages the benefits of a multithreaded environment to distribute the gradient calculation during training. The training instances in a minibatch are shared evenly across the number of available threads (specified by the user via the `--threads` parameter). In our gradient descent implementation, we use adaptive learning (Duchi et al., 2011) to converge to a better set of parameters.

We rely on `Eigen`, a high-level `C++` library for linear algebra, to speed up the matrix and vector operations involved in training and querying our models. Where possible, we group together multiple similar operations to further speed up the computations.

Finally, we speed up the process of tuning the translation system feature weights by taking advantage of the fact that the development corpus is decoded for several iterations with different weights. As a result, the n-grams scored by the language model often repeat themselves over a number of iterations. We maintain sentence-specific caches mapping n-grams to language model probabilities which are persistent between consecutive iterations of the tuning algorithm. A persistent cache is loaded from disk when a sentence is received as input and saved back to disk when the system has finished decoding the sentence. This optimization massively speeds up the process of tuning the translation system and can be enabled via the `--persistent-cache` flag in the decoder configuration file.

5. Experiments

In this section, we provide experimental results to illustrate the strengths of our language modelling framework. We report perplexities and improvements in the BLEU score when the language model is used as an *additional* feature in the decoder. We also report the training times for stochastic gradient descent and noise contrastive estimation. Finally, we compare the average time needed to decode a sentence with our language modelling feature against a standard system using only an efficient implementation of a backoff n-gram model.

In our experiments, we used the `europarl-v7` and the `news-commentary-v9` French-English data to train a hierarchical phrase-based translation system (`cdec`). The corpus was tokenized, lowercased and filtered to exclude sentences longer than 80 words or having substantially different lengths using the preprocessing scripts available in `cdec`[4]. After preprocessing, the training corpus consisted of 2,008,627 pairs of sen-

[4]We followed the indications provided here: `http://www.cdec-decoder.org/guide/tutorial.html`

Model	Training algorithm	Training time (hours)	Perplexity	BLEU
KenLM	-	0.1	267.814	24.75
FactoredLM	SGD	34.1	226.44	25.2
FactoredLM	NCE 1	9.4	258.623	25.25
FactoredLM	NCE 10	12.5	245.748	25.06
FactoredLM	NCE 50	18.1	241.481	25.23
DirectFactoredLM	SGD	42.0	210.275	25.46

Table 1. Training time, perplexities and BLEU scores for various models.

tences. The corpus was aligned using `fast_align` (Dyer et al., 2013) and the alignments were symmetrized using the `grow-diag-final-and` heuristic. We split the `newstest2012`[5] data evenly into a development set and a test set, by assigning sentences to each dataset alternatively. The translation system is tuned on the development set using MIRA. We report average BLEU scores over 3 MIRA runs.

The baseline system includes an efficient implementation (Heafield, 2011) of a 5-gram language model (`KenLM`). The language models are trained on the target side of the parallel corpus, on a total of 55,061,862 tokens. Before training the neural language models, singletons are replaced with a special `<unk>` token. The neural language model vocabulary consists of 57,782 words and is factored into 240 classes using Brown clustering. In our experiments, we set the order of the neural language models to 5, the dimensionality of the word representations to 200 and make use of diagonal contexts. The standard factored language model is labelled with `FactoredLM`. `DirectFactoredLM` is an extension incorporating direct n-gram features. In our experiments, we define a feature function for every n-gram ($n \leq 5$) observed at least 3 times in the training corpus. The feature weights are hashed into a collision store with a capacity of 5 million features.

Table 1 summarizes the results of our experiments. We indicate the algorithm used to train each neural language model. Stochastic gradient descent is denoted by SGD, while noise contrastive estimation is denoted by NCE and followed by the number of noise samples used for estimating the gradient for each data point. In both cases, the gradient optimization was distributed over 8 threads and the minibatch size was set to 10,000 data points. We note that noise contrastive estimation leads to models with higher perplexities. However, that has no effect on the overall quality of the translation system, while massively reducing the training time of the neural language models. Overall, we observe a BLEU score improvement of 0.7 when a factored language model with direct n-gram features is used in addition to a standard 5-gram language model.

[5]The corpus is available here: `http://www.statmt.org/wmt14/translation-task.html`

Model	Decoding time (seconds)
`KenLM`	0.447
`FactoredLM`	3.356
`DirectFactoredLM`	6.633

Table 2. Average decoding speed.

Table 2 shows the average decoding speed with our neural language modelling features. The average decoding time is reported on the first 100 sentences of the development set. Overall, the neural language models slow down the decoder by roughly an order of magnitude.

In conclusion, this paper presents an open source implementation of a neural language modelling toolkit. The toolkit provides techniques for speeding up training and querying language models and incorporates direct n-gram features for better translation quality. The toolkit facilitates the integration of the neural language models as a feature in the beam search of the `cdec` and `Moses` decoders. Although our language modelling features slow down the decoders somewhat, they guarantee that the probabilities used to score partial translation hypotheses are properly normalized.

Bibliography

Auli, Michael and Jianfeng Gao. Decoder integration and expected bleu training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pages 136–142, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Botha, Jan A. and Phil Blunsom. Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML '14)*, Beijing, China, 2014.

Brown, Peter F., Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Chelba, Ciprian, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, 2013.

Chen, Stanley F. and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

Duchi, John, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '13)*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Goodman, Joshua. Classes for fast maximum entropy training. *CoRR*, 2001.

Heafield, Kenneth. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT '11)*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Liang, P. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

Mikolov, Tomas, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. Strategies for training large scale neural network language models. In *Proceedings of the 2011 Automatic Speech Recognition and Understanding Workshop*, pages 196–201. IEEE Signal Processing Society, 2011a.

Mikolov, Tomas, Stefan Kombrink, Anoop Deoras, Lukas Burget, and Jan Cernocky. Rnnlm - recurrent neural network language modeling toolkit. In *Proceedings of the 2011 Automatic Speech Recognition and Understanding Workshop*, pages 1–4. IEEE Signal Processing Society, 2011b.

Mnih, Andriy and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pages 641–648, Corvallis, OR, USA, 2007.

Mnih, Andriy and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning (ICML '12)*, pages 1751–1758, Edinburgh, Scotland, 2012.

Morin, Frederic and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS '05)*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

Schwenk, Holger. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

Schwenk, Holger. Continuous-space language models for statistical machine translation. *Prague Bulletin of Mathematical Linguistics*, 93:137–146, 2010.

Stolcke, Andreas. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 901–904, 2002.

Vaswani, Ashish, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Address for correspondence:
Paul Baltescu
paul.baltescu@cs.ox.ac.uk
Department of Computer Science
University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD,
United Kingdom

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 93–104

Multilingual Dependency Parsing: Using Machine Translated Texts instead of Parallel Corpora

Loganathan Ramasamy, David Mareček, Zdeněk Žabokrtský

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

This paper revisits the projection-based approach to dependency grammar induction task. Traditional cross-lingual dependency induction tasks one way or the other, depend on the existence of bitexts or target language tools such as part-of-speech (POS) taggers to obtain reasonable parsing accuracy. In this paper, we transfer dependency parsers using only approximate resources, i.e., machine translated bitexts instead of manually created bitexts. We do this by obtaining the the source side of the text from a machine translation (MT) system and then apply transfer approaches to induce parser for the target languages. We further reduce the need for the availability of labeled target language resources by using unsupervised target tagger. We show that our approach consistently outperforms unsupervised parsers by a bigger margin (8.2% absolute), and results in similar performance when compared with delexicalized transfer parsers.

1. Introduction

Inducing dependency structures has been an important topic of research within the parsing community for many years. Dependency parsers that can produce dependency structures for novel sentences often rely on manually constructed treebanks for training the parsers. Unlike other annotation tasks such as POS tagging, treebank annotation is much more complex and expensive. Zeman et al. (2012) identified the availability of treebanks for 30 languages. However, still majority of languages do not have treebanks. Thus, inducing treebanks for languages that have small or no training data is definitely a challenging task. Though fully unsupervised dependency parsing approaches (Mareček and Straka, 2013; Spitkovsky et al., 2013, Blunsom and

 Corresponding author: ramasamy@ufal.mff.cuni.cz
Cite as: Loganathan Ramasamy, David Mareček, Zdeněk Žabokrtský. Multilingual Dependency Parsing: Using Machine Translated Texts instead of Parallel Corpora. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 93–104. doi: 10.2478/pralin-2014-0017.

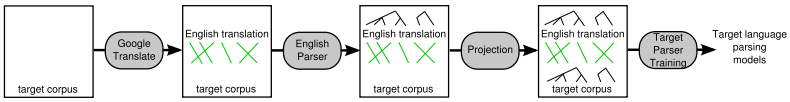


Figure 1. Schematic depiction of how the target parse trees are obtained

Cohn, 2010) are quite attractive for they don't require any hand annotated data for training, their quality is still lower than other class of approaches mainly known as *cross-lingual syntactic transfer* techniques. This is a useful alternative in transferring syntactic knowledge from one or more languages.

Hwa et al. (2005) used parallel corpus and word alignments to project English parse trees to Spanish and Chinese. They have also used small number of language specific transformation rules to reduce projection errors due to different annotation choices in the treebanks. Most of the earlier transfer based approaches (Ganchev et al., 2009; Kuhn, 2004) heavily rely on bitexts or some other target[1] language resources (such as POS taggers).

Transfer based techniques such as Zeman and Resnik (2008) and McDonald et al. (2011b) decouple this target language resource requirement by directly parsing target sentences via delexicalized source parser (trained with source POS tag sequence). Delexicalized parsing depends only on target POS tagger which uses the same POS tagset as the source language tagger. Augmenting delexicalized parsers by cross-lingual clusters (Täckström et al., 2012), bilingual lexicon (Durrett et al., 2012) and target adaptation techniques (Täckström et al., 2013) further improved the delexicalized parsers. Most of the recent works on transfer parsers sought to reduce POS annotation differences between source and target languages by mapping to a common coarse-grained tagset (Petrov et al., 2012). Addressing annotation differences at the structural level in transfer parsers is still an open problem, though there are some early attempts such as Smith and Eisner (2009) and more recently Zeman et al. (2012); McDonald et al. (2013) through treebank harmonization and by common annotation standards.

It has been well established from previous works that the availability of bitexts or target POS taggers (or both) is very crucial for transferring dependency parsers from one or multiple source languages. Imagine a situation where we don't have direct access to bitexts or a target POS tagger but only to a translation system from a resource-poor (RP) language to a resource-rich (RR) language. This presents an interesting scenario for the existing transfer based approaches. In this paper, we propose to combine bitexts obtained from machine translation and target POS obtained from

[1] *target* refers to language(s) for which we would be interested in inducing dependency parser. *source* in other words refers to language(s) from which we will transfer the dependencies. It is also assumed that target languages are resource-poor whereas source languages are resource-rich.

unsupervised clusters to obtain transfer parsers. We use MT system to translate target language texts to resource-rich source language texts (for which parsers and taggers are available). Our overall approach is similar to Hwa et al. (2005) and McDonald et al. (2011b), but the main difference lies is in the nature of bitexts we use for transferring the parsers.

Later in the results section, we show that this approach outperforms state-of-the-art unsupervised approaches even though we use only approximate bitexts for our transfers.

2. Dependency Transfer With Machine Translated Texts

The heart of our approach lies in how we obtain bitexts and target POS taggers (for resource-poor languages), which are crucial for transfer parsers. Unlabeled data is available in plenty even for resource-poor languages. We obtain both the resources from unlabeled target texts only. For word aligned bitexts, we first translate target texts into English via an MT system, in our case Google Translate API.[2] Our system is single source, i.e., all our experiments are carried out with English as a source language against a variety of target languages. For word alignments, we use alignment links provided by the MT system. If word alignments are not provided by the MT system, then any automatic word aligners can be used to obtain the word alignments. We parse translated English source texts using the parsing model trained on the English treebank from CoNLL shared task (Nivre et al., 2007). Our overall approach is depicted in Figure 1.

We obtain fully connected target language parse trees by projecting English parse trees onto target sentences via word alignment links. Before projection, we initialize the target tree by connecting all the target tokens to the default root node of the tree. The projection algorithm then starts from the source root and visits all the source nodes in a pre-order fashion while making adjustments to parents in the target tree.

In the case of 1-M alignments, we first determine the head of the chunk on the target side and connect the remaining members of that chunk to the chunk head. We make a simplistic assumption about chunk head: i.e., we consider the last member (its absolute position should also be higher than other members) to be the chunk head. Unlike Hwa et al. (2005), we do not add empty nodes on the target in the case of determining target parents for 1-M and unaligned source nodes. We do that so for simplifying the evaluation of target trees.

We use the projected target trees to train various parsing models. Previous works have mostly relied on using target POS taggers for training the parsers. McDonald et al. (2011b) demonstrated that POS information alone carries much of the syntactic information, thus making use of target POS taggers considerably improved the accuracy of the target parse trees. To make our work applicable in realistic scenarios, we

[2]Google Translate API – `https://developers.google.com/translate/`

#	ar	bg	ca	cs	da	de
Sentences	3.0K	13.2K	14.9K	25.6K	5.5K	38.0K
Tokens	116.8K	196.2K	443.3K	437.0K	100.2K	680.7K
Train/Test(%)	96/4	97/3	88/12	99/1	94/6	95/5
#	el	es	et	fi	hi	hu
Sentences	2.9K	16.0K	1.3K	4.3K	13.3K	6.4K
Tokens	70.2K	477.8K	09.5K	58.6K	294.5K	139.1K
Train/Test(%)	93/7	90/10	90/10	90/10	91/9	94/6
#	it	nl	pt	sl	sv	tr
Sentences	3.4K	13.7K	9.4K	1.9K	11.4K	5.9K
Tokens	76.3K	200.7K	212.5K	35.1K	197.1K	69.7K
Train/Test(%)	93/7	97/3	97/3	79/21	97/3	95/5

Table 1. Target language treebank texts that are translated to English using MT system

induce target POS information using unsupervised techniques. Unsupervised POS tagging is arguably less complex than inducing tree structures and previous works on unsupervised POS techniques (Blunsom and Cohn, 2011; Clark, 2000) have proven to be effective even in practical applications. In this work, we use unsupervised target POS tags instead of supervised or universal POS tags, but for the sake of comparison, we also provide results with supervised and universal POS tags. We experiment with various tagset size and show results for the tagset size that gives the best average accuracy on target languages. Our approach can be used within the transfer framework for languages that lack even POS taggers, thus making the approach suitable for languages that do not have any labeled target language resources.

3. Experiments

We use 18 treebanks (see Table 1) for most of our experiments. In certain experiments, we show results for a subset of those languages for comparison with other works. For training/testing, we use the same data split as described in Zeman et al. (2012). The target language treebanks we use mostly come from past CoNLL shared tasks (2006, 2007 and 2009). For Hindi, we have used the latest version (ICON 2012) of the treebank instead of the version mentioned in Zeman et al. (2012). All our results show only *unlabeled attachment score (UAS)* accuracies – similar to other works in the field.

3.1. Projection

The schema of the projection procedure is depicted in Figure 1. It consists of four steps:

Lang.	Baseline		UDP	Projection (+ reparsing)					sup
	left	right		unsup40	dir proj	univ	sup	gold	
ar	5.2	**58.8**	34.1	51.8	40.3	56.1	58.3	60.0	74.2
bg	17.9	38.8	56.7	46.4	41.4	**55.3**	53.6	56.3	81.5
ca	24.7	28.8	24.6	56.9	46.0	-	**59.0**	60.2	87.6
cs	24.1	28.9	55.0	45.4	51.4	54.7	**58.4**	62.3	74.9
da	13.2	**47.9**	42.0	41.6	36.9	41.6	42.1	43.1	79.8
de	24.2	18.9	43.5	46.4	43.0	47.8	**48.7**	50.1	84.2
el	32.0	18.5	30.9	49.2	52.0	59.2	**65.4**	65.2	78.5
es	24.7	29.0	36.3	56.9	47.0	-	**58.2**	59.0	88.1
et	34.1	17.4	**63.8**	54.8	58.0	-	58.3	66.0	72.9
fi	39.3	13.6	36.7	37.1	43.1	-	**39.5**	46.2	60.7
hi	24.4	27.3	15.6	24.9	24.6	-	**28.0**	28.3	75.1
hu	42.8	5.3	34.7	43.4	41.1	48.2	**51.2**	53.2	75.2
it	23.0	37.4	49.7	55.6	53.1	53.7	**59.3**	62.0	80.5
nl	27.9	24.7	27.6	**60.9**	53.2	-	59.3	61.4	76.7
pt	25.8	31.1	39.8	61.8	56.4	62.2	**62.9**	66.5	84.2
sl	24.4	26.6	45.9	44.4	50.4	45.7	**53.1**	56.6	76.7
sv	25.9	27.8	49.1	53.9	48.2	**56.8**	54.3	55.8	82.7
tr	**65.1**	2.0	42.3	46.0	51.4	-	57.2	57.0	75.5
avg	27.7	26.8	40.5	48.7	46.5	*52.8*	**53.7**	56.1	78.3

Table 2. UAS for baselines, supervised/unsupervised parsers and various projected parsing models. Resource requirements for projection under various settings: unsup40 *requires bitext, source parser and unsupervised target tagger;* dir proj *requires only bitext;* univ *requires bitext, source parser, and universal source/target taggers;* sup *(column 8) requires bitext, source parser, and target tagger; the resource requirements for* gold *is similar to* sup *but requires gold POS during testing.* sup *(column 10) is a regular supervised parser results.* UDP *shows results for unsupervised dependency parsing. Numbers in* ***bold*** *indicate the best accuracy for each row (excluding columns 9 and 10).*

1. The target language corpus is translated to English using Google Translate API v1. The API provides also alignment links, which will be used for projection. The translated English sentences are then tokenized and the original alignment links are adjusted.[3] For the translation task, we confined ourselves to translating only the treebank data, but much larger texts can be used in the future.

[3]For instance, when a punctuation is separated from a form on the English side, we link the separated punctuation to the corresponding punctuation on the treebank data.

2. English sentences are tagged by the Morce tagger (Spoustová et al., 2007) and parsed by the MST parser (McDonald et al., 2005). For parsing English, we used the parser model trained on the version of Penn treebank supplied during the CoNLL 2007 shared task (Nivre et al., 2007).
3. English dependency trees are projected to the target language sentences (only the training part of the treebank) using the alignment links.
4. Three target parser models are trained (using MST parser with 2^{nd} order and non-projective setting) on the projected target corpus with different POS annotations (next subsection).

3.2. Training with different POS tags

To tag target test data, we train two supervised taggers and one unsupervised tagger on the training section of the target treebanks.

- **Supervised POS:** We train Stanford tagger (Toutanova and Manning, 2000) on the training section of the treebanks.
- **Universal POS:** We first convert the annotated training data to universal POS tags (Petrov et al., 2012) and train the tagger on it.
- **Unsupervised POS tagger:** We use unsupervised hidden Markov model (HMM) POS tagger by Blunsom and Cohn (2011). Not knowing which tagset size is suitable for the projected treebanks, we obtain POS tags for different arbitrary tagset size: 20, 40, 80 and 160. Besides the training and testing parts of the treebanks[4], we used additional monolingual texts from W2C Wikipedia corpus (Majliš and Žabokrtský, 2012) to enlarge the size of the data to one million words for each language.

3.3. Direct transfer of delexicalized parser

We train delexicalized English parsers under two settings. In the first setting, we convert the POS tags of CoNLL 2007 English data into universal POS tags using the mapping provided by Petrov et al. (2012), strip all the word forms and train the parser. In the second setting, we first tag the English translations from Google Translate using the Morce tagger (Spoustová et al., 2007), convert them to universal POS tags and after stripping all the word forms, we train delexicalized parser on them. We obtain target universal POS tags using the universal POS tagger trained from the training part of the target treebanks. So, the main difference between McDonald et al. (2011b) and our approach is that they obtained target POS tags by POS projection from English, whereas we used POS information from target language treebanks and trained uni-

[4]The unsupervised POS tagger by Blunsom and Cohn (2011) does not produce a trained POS model. Given an unlabeled data and tagset size, the tagger produces unsupervised tags for the unlabeled data. Our unlabeled data included the training/testing part of the treebanks and some additional monolingual corpus.

versal POS taggers on them. At the moment, our experiments on unsupervised POS taggers deal with varying tagset size, and it would also be interesting in the future to make a comparison with different unsupervised approaches such as unsupervised POS projection (Das and Petrov, 2011).

3.4. Unsupervised parsing

To compare the projection results with a completely unsupervised approach, we used the software for dependency grammar induction by Mareček and Straka (2013).[5] We run the experiments in the same manner as they described for all our testing languages.

4. Results

Our major results are presented in Table 2 and 3. The left and right baselines in Table 2 indicate that some languages have a strong preference for either left or right branching.

- *sup* presents UAS scores from a supervised parser trained on the training portion of the target treebanks.
- *UDP* presents UAS scores achieved in unsupervised dependency parsing (Mareček and Straka, 2013) on the test portion of the target treebank data.
- Projected results under different settings
  - *unsup40* presents UAS scores for parsing the test data with unsupervised POS tags (tagset size 40). We also experimented with different tagset size: 20, 80 and 160. We chose the tagset size that gave the best average accuracy.
  - *dir proj* shows UAS scores from directly projecting translated English test data onto target test sentences.
  - *univ*: test data is tagged by universal POS tagger before parsing.
  - *gold*: test data is tagged with gold POS tags before parsing.
  - *sup*: test data is tagged by supervised POS tagger before parsing.

For the sake of comparison, we reproduce delexicalized parser results on our data with two settings: (i) delexicalized parser trained on the POS tags of CoNLL 2007 (Nivre et al., 2007) English data and (ii) delexicalized parser trained on the POS tags of English translations obtained from the MT system. The results are shown in Table 3. For both settings, we obtain target POS tags in a supervised manner. We also provide delexicalized parser results from McDonald et al. (2011a). One intriguing aspect of these results is that delexicalized parsers obtained from the machine translated texts perform better than the delexicalized parser obtained from the CoNLL 2007 data (48.4 vs. 47.8). *McD 2011* has better overall results compared to our delexicalized parsers. We attribute this to difference in parser training parameters as well as the usage of

[5] http://ufal.mff.cuni.cz/udp/

Lang.	Delex CoNLL	Delex GT	*McD 2011*
ar	29.1	31.9	-
bg	52.8	51.4	-
cs	35.6	35.4	-
da	44.6	39.7	*45.5*
de	47.5	48.1	*47.5*
el	59.0	60.5	*65.2*
es	-	-	*52.4*
hu	45.0	46.2	-
it	52.9	57.2	*56.3*
nl	-	-	*66.5*
pt	65.4	63.5	*67.7*
sl	36.5	44.2	-
sv	57.7	54.5	*59.7*
avg	47.8	48.4	*57.6*

Table 3. Delexicalized direct transfer parsers comparison (unlabeled attachment score – UAS). Delex CoNLL – *delexicalized parser trained on the CoNLL 2007 (Nivre et al., 2007) data.* Delex GT – *delexicalized parser trained on the POS tags of English translations obtained using Google Translate API.* McD 2011 – *results from McDonald et al. (2011a).*

POS projection instead of a supervised POS tagger. When we make an overall comparison (Tables 2 & 3), the advantages of training the supervised taggers (both original and universal POS) are clearly visible. However, *unsup40* outperforms *UDP* by 8.2% absolute UAS score, and also gives slightly better results with respect to *delex CoNLL* and *delex GT*. Remember, both *delex CoNLL* and *delex GT* use supervised target taggers, that means, *unsup40* does not use any labeled target resources, but still performs better than other strategies. This suggests that, the syntactic projection which crucially relies on bitexts can still be beneficial even in the absence of high quality bitexts.

5. Discussion

From the results given in the previous section we can see that the average UAS of parsing models trained on the projected trees outperforms left and right baselines, while it is well below the supervised parser UAS, which was expected. In most cases it is above the unsupervised parser UAS, which was not guaranteed before performing our experiments. The performance difference between using hand-designed POS tags of the original treebanks and unsupervised tags (induced from large unannotated data) is surprisingly small.

Many techniques have been proposed in the past to address variety of resource poor scenarios, to which we would like to add one more scenario, the availability of an

MT system from a resource-poor language to a resource-rich language. To summarize, if one wants to parse a text in a language, the following procedure typically leads to the best performance:

1. If there is a treebank for the language, use it for training a supervised parser.
2. If there is no treebank, but a POS tagger exists, then develop a conversion to the universal tagset and use the delexicalized parser.
3. If the tagger is not available, but an MT system from the language to English exists, then use our approach.
4. If none of the previous steps is applicable, then use unsupervised parsing.

There are published treebanks for around 40 languages, and there are more than 70 languages supported by Google Translate. So one can expect around 30 languages for which either step 2 or 3 leads to state-of-the-art results. Given the quick growth of the number of languages covered by Google Translate, we believe that our approach will be viable for more and more languages.

In addition, our approach might be helpful for languages whose treebanks and taggers are available only under very restrictive licenses.

6. Conclusion

In this paper, we have considered the dependency induction task for a specific resource-poor scenario in which only MT system from a resource-poor language to a resource-rich language is available. We have used machine translated bitexts as a substitute for high quality bitexts, and used source language MT outputs and target language texts as a basis for projection-based transfer approach. The experimental results show that, in realistic scenarios, the projection-based transfer can be combined with unsupervised target POS tagger to achieve better parsing performance than unsupervised parser and similar performance as delexicalized transfer parsers. In the future, it would be interesting to compare transfer parsers induced from human translated bitexts and machine translated bitexts and ascertain whether MT outputs can be used as a substitute for tasks which require bitexts.

Acknowledgements

This research has been supported by the grant no. GPP406/14/06548P of the Grant Agency of the Czech Republic. This work has been using language resources developed and/or stored and/or distributed by the LINDAT/CLARIN project of the Ministry of Education of the Czech Republic (project LM2010013).

Bibliography

Blunsom, Phil and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural*

Language Processing, EMNLP '10, pages 1204–1213, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Blunsom, Phil and Trevor Cohn. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Clark, Alexander. Inducing syntactic categories by context distribution clustering. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, CoNLL '00, pages 91–94, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Das, Dipanjan and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 600–609, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002549`.

Durrett, Greg, Adam Pauls, and Dan Klein. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D12-1001`.

Ganchev, Kuzman, Jennifer Gillenwater, and Ben Taskar. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 369–377, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL `http://dl.acm.org/citation.cfm?id=1687878.1687931`.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:311–325, 9 2005. ISSN 1469-8110. doi: 10.1017/S1351324905003840. URL `http://journals.cambridge.org/article_S1351324905003840`.

Kuhn, Jonas. Experiments in parallel-text based grammar induction. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1218955.1219015. URL `http://dx.doi.org/10.3115/1218955.1219015`.

Majliš, Martin and Zdeněk Žabokrtský. Language richness of the web. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Mareček, David and Milan Straka. Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Langauge Technology Con-*

ference and Conference on Empirical Methods in Natural Language Processing (HTL/EMNLP), pages 523–530, Vancouver, BC, Canada, 2005.

McDonald, Ryan, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 62–72, Stroudsburg, PA, USA, 2011a. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145440.

McDonald, Ryan, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK., July 2011b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1006.

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-2017.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Petrov, Slav, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Smith, David A. and Jason Eisner. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 822–831, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-62-6. URL http://dl.acm.org/citation.cfm?id=1699571.1699620.

Spitkovsky, Valentin I., Hiyan Alshawi, and Daniel Jurafsky. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Spoustová, Drahomíra, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In *ACL '07: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 67–74, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

Täckström, Oscar, Ryan McDonald, and Jakob Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1052.

Täckström, Oscar, Ryan McDonald, and Joakim Nivre. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N13-1126.

Toutanova, Kristina and Christopher D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70, Hong Kong, October 2000.

Zeman, Daniel and Philip Resnik. Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India, 2008. Asian Federation of Natural Language Processing, International Institute of Information Technology.

Zeman, Daniel, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To Parse or Not to Parse? In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Address for correspondence:
Loganathan Ramasamy
ramasamy@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014 105–126

An Interplay between Valency Information and Reflexivity

Václava Kettnerová, Markéta Lopatková, Jarmila Panevová

Charles University in Prague, Faculty of Mathematics and Physics

A Response to R. Wagner's Contribution: A Case of Collision in Principles of Language Description?

Abstract

A language description based on a formally defined framework has many advantages: The possibility to check the inner consistency of the model as well as the possibility of comparison with other models or with pure descriptive approaches belong to its main priorities.

Roland Wagner's contribution published in the last issue of this journal – focusing (among other ideas) on the role of Czech reflexives – presents several critical remarks concerning the Functional Generative Description. These remarks represent a good challenge for the authors developing this model to fill empirical gaps and to make clear some theoretical presuppositions concerning valency frames of verbs and their respective reflexive counterparts that are primarily addressed by Roland Wagner's critical survey.

1. Introduction

Roland Wagner's (RW in sequel) account how the Czech reflexives *se/si* are analyzed within the theoretical framework of the Functional Generative Description (FGD in sequel) – summarized in his article (Wagner, 2014) as Principle 2 – is correct: (i) Those reflexives that are either parts of a lexical entry of a verb lemma, see examples (1) and (2), or those that are grammatical markers of generalized Actors, see (3), are considered reflexive particles, while (ii) the reflexives *se/si* representing the valency complementation coreferential with the subject of the sentence (or with

 Corresponding author: lopatkova@ufal.mff.cuni.cz
Cite as: Václava Kettnerová, Markéta Lopatková, Jarmila Panevová. An Interplay between Valency Information and Reflexivity. The Prague Bulletin of Mathematical Linguistics No. 102, 2014, pp. 105–126.
doi: 10.2478/pralin-2014-0018.

another embedded subject), see examples (4) and (5), are interpreted in FGD as reflexive pronouns expressing the respective syntactic function in the sentence.

(1) *Jan se smál.*
John refl laughed
En. John was laughing.

(2) *Dny se v létě prodlužují.*
daytime refl in summer prolong
En. Daytime is becoming longer in summer.

(3) *Termín odeslání článku se prodloužil.*
the deadline for submitting a paper refl extended
En. The deadline for submitting a paper was extended.

(4) *Petr se každé ráno myl studenou vodou.*
Peter refl every morning washed with cold water
En. Peter washed himself with cold water every morning.

(5) *Matka nařídila Petrovi umýt se.*
mother ordered Peter wash refl
En. The mother ordered Peter to wash himself.

Further, the overall claim that according to FGD "differences in valency frames correlate with differences in lexical meaning […]" (Principle 1 in RW's text) reflects one of the main ideas of the valency theory of verbs in FGD and its consequence (postulated by the author) that a single lexical unit of a verb cannot be assigned with more than one valency frame is entirely acceptable. However, the notion of (grammatical) meaning and its reflection in valency frames of verbs require clarification.

In valency lexicons elaborated within FGD – henceforth we (similarly as RW) refer to the valency lexicon of Czech verbs, VALLEX[1] – valency frames are modeled as a sequence of valency slots; each slot stands for one complementation and consists of:

- the semantic relation to its governing verb (labeled by a functor),
- the information on the type of valency complementation with respect to its obligatoriness, and
- possible morphemic forms which are specified for the complementations whose form is prescribed by the verb.

However, in the strict sense, only the information on the number and the type of valency complementations is relevant for grammatically structured meaning (the tectogrammatical layer of FGD) of the verb; the information on possible morphemic form(s) of a valency complementation characterizes its surface syntactic expression. As it is the correlation between functors and morphemic forms that determines the

[1]The Valency Lexicon of Czech Verbs, VALLEX, is available at `http://ufal.mff.cuni.cz/vallex/2.5`, or in the published version (Lopatková et al., 2008).

meaning of a lexical unit, both types of information are encoded in valency frames.[2] Let us stress that in the FGD based valency lexicons, the morphemic expressions of valency complementations are limited to the usage of a lexical unit of a verb in *active*, *nonreflexive*, *nonreciprocal* constructions, see esp. (Lopatková et al., 2008).

Let us now repeat the case of seeming collision of Principles 1 and 2 as it was exemplified by RW in his article by the verb *vnímat* 'to see, to perceive', see examples (6)–(8) ((2)–(4) in his paper). RW demonstrates the change of the morphological form of the participant EFFect from *jako*+Acc (in (6)) into *jako*+Nom when PATient is lexically expressed by the clitic form of the reflexive pronoun (in (7)) (while the morphemic form *jako*+Acc of EFF is indicated in the VALLEX lexicon, the morphemic expression *jako*+Nom is missing there, see (20)). Then he infers that – on the basis of Principle 1 – the change in morphosyntactic form of EFFect implies the necessity of two different lexical units for the two usages of the verb *vnímat* 'to see, to perceive' in examples (6) and (7). However, in accordance with Principle 2, the usages of the verb *vnímat* 'to see, to perceive' in sentences (7) and (8) represent the same lexical unit since the non-clitic reflexive together with the clitic reflexive forms a single morphological paradigm of the pronoun.

(6) *Vnímá syna jako soka.* (RW (2))
(he) sees son$_{Acc}$ as a rival$_{jako+Acc}$
En. He sees his son as a rival.

(7) *Sám se vnímá*
himself$_{Nom}$ refl$^{clitic}_{Acc}$ (he) sees
jako síla „ochraňující divadlo". (RW (3), SYN2005)
as a force$_{jako+Nom}$ "sheltering theatre"
En. He sees himself as a force "sheltering theatre".

---

[2] Another possibility is to accept the concept of structural and lexical cases, as it is proposed by Karlík (2000), and limit the information on possible morphemic expression(s) only to valency complementations expressed by lexical cases. However, there are several issues undermining such solution.

(i) From a theoretical point of view, non-prototypical changes of structural cases should be described and taken into account when operating with this dichotomy (compare the prototypical change of Acc into Nom in the passive construction with the non-prototypical change of the Acc into Dat in the nominalization: *Prezident vyzval*$_{\text{active}}$ *premiéra*$_{\text{acc}}$ *k rezignaci* → *Premiér*$_{\text{nom}}$ *byl vyzván*$_{\text{passive}}$ *k rezignaci prezidentem*, however, *Prezident vyzval*$_{\text{active}}$ *premiéra*$_{\text{acc}}$ *k rezignaci* → *výzva (prezidenta) premiérovi*$_{\text{dat}}$) (Kolářová, 2010).

(ii) From a lexicographic point of view, structural cases can be omitted on condition that there exists an elaborated classification of verbs allowing for the prediction of changes of structural cases in different syntactic contexts. For the time being, we are not aware of a sophisticated reliable classification of Czech verbs that could be adopted for the lexicon.

Technically, as the information on the (in)transitivity (and maybe other features) of individual lexical units should be recorded for each lexical unit, we opt for the equivalent information on the nominative and accusative complementation.

(8) *Karel IV. vnímá sebe jako vyvoleného třetího krále.* (RW (4), SYN2005)
Charles IV sees $\text{refl}^{non\text{-}clitic}_{Acc}$ as chosen third king$_{jako+Acc}$
En. Charles IV sees himself as the chosen third king.

Let us point out that – with respect to the clarified interpretation of Principle 1 (see above) – we do not face in fact a collision of the two principles (as the morphemic changes related to reflexivity are not considered relevant for delimiting a new lexical unit of a verb). What we must in fact cope with is a gap in the description of changes in valency structures of Czech verbs as described for VALLEX, see esp. (Kettnerová and Lopatková, 2009), (Kettnerová et al., 2012a).[3]

Though the number of the verbs concerned is very limited[4] (despite the fact that this change is exhibited by relatively frequent verbs, it is very rare in corpus data, see the Appendix for the statistics), RW's remarks remind the authors of the VALLEX lexicon that the changes in morphosyntactic expressions of valency complementations conditioned by a broader syntactic context have not yet been described exhaustively enough.

In the next sections, we demonstrate that the linguistic phenomenon addressed by RW can be easily integrated in the descriptive apparatus of FGD. In the following section, an enhanced version of FGD that takes a close interplay of lexical and grammar information into account is introduced (Section 2). Further, the application of the principles of the enhanced version of FGD on the analysis of the addressed phenomena is presented in Section 3. Finally, theoretical considerations concerning reflexivity are addressed in Section 4.

2. Enhanced FGD: grammar and lexical components

Contemporary linguistic frameworks are based on the division of labor between lexical and grammar components; each of which gives greater or lesser prominence either to a lexical, or to a grammar part of the linguistic description. Let us point to Chomskyan generative grammar and the Meaning Text Theory as two illustrative examples of almost opposing tendencies: in the former, the key role is performed by

[3]We would like to express our gratitude to Richard Wagner for pointing out this specific change in valency structure of verbs related to reflexivity.

[4]RW found 21 lexical units of verbs contained in VALLEX in total leading to the seeming conflict between Principle 1 and Principle 2. We agree with his findings (with the exception of verbs *angažovat*$_1$, *brát*$_7$, *udržovat/udržet*$_3$ and *fotografovat*$_1$, which do not meet the required pattern; on the other hand, we can add other verbs as *stanovit*$_2$, *přijímat/přijmout*$_5$, and *přijímat/přijmout*$_9$; see the Appendix for the full list of affected verbs in VALLEX). We can realize that the analyzed phenomenon is quite rare – for most verbs it concerns less than 1% of their occurrences in CNC (only for the verbs *prezentovat* 'to present' and *označovat*^{impf} 'to declare, to call' the rough estimation exceeds 3.2% and 2.4%, respectively; for three other verbs the estimation reaches 1–2% (for one of their aspectual counterparts)).

a grammar component, while the latter relies esp. on a thoroughly elaborated lexical component.

Since the original proposal of FGD (Sgall, 1967), both grammar and lexical modules have been taking into account; however, the main focus has been laid on grammar, esp. syntactic description of a language (Sgall et al., 1986). The importance of a lexical module has been growing since the extensive application of the theoretical results on corpus data during the work on the Prague Dependency Treebank (Hajič et al., 2006). At present, there are several lexicons elaborated within the theoretical framework of FGD: PDT-VALLEX (Urešová, 2011), VALLEX (Lopatková et al., 2008), EngVALLEX (Cinková, 2006; Šindlerová and Bojar, 2009).

Recently, a special attention has been devoted to linguistic phenomena on the lexicon-grammar interface, requiring a close interplay between grammar and lexical modules: e.g., grammatical diatheses, reflexivity and reciprocity. They represent more or less productive syntactic operations that are regular enough to be described by formal syntactic rules. Although general semantic and syntactic observations can be usually made about these phenomena, their applicability is still lexically conditioned and as such has to be recorded in lexical entries of relevant verbs in a lexicon, see esp. (Kettnerová and Lopatková, 2009, 2011; Kettnerová et al., 2012a,b) and (Panevová and Ševčíková, 2013).

As a result, the valency characteristics of lexical units are partially stored in a valency lexicon, partially they are derived by grammatical rules (closely cooperating with the lexicon). Let us exemplify this cooperation on the example of the passive diathesis:

(9) *Stát zvýhodní podnikatelské záměry v hospodářsky problémových oblastech vyššími podporami a speciálními programy.* (PDT, modified)
En. The government makes business plans in business problem regions favorable by higher grants and special programs.

(10) *Podnikatelské záměry v hospodářsky problémových oblastech jsou zvýhodněny vyššími podporami a speciálními programy.* (PDT)
En. Business plans in business problem regions are made favorable by higher grants and special programs.

(11) *zvýhodnit*$_1$ 'to make favorable' … $\mathsf{ACT}_1\ \mathsf{PAT}_4\ \mathsf{MEANS}_7^{\mathsf{typ}}$
-diat: pass, deagent, res-být, res-mít

First, the valency frame of the verb *zvýhodnit*$_1$ 'to make favorable' consists of two valency complementations, ACTor and PATient. The VALLEX lexicon contains information on possible morphemic forms of valency complementations for the active usage of the verb, as in (9) – namely ACTor in nominative and PATient in accusative,

see (11);[5] moreover, the lexicon entry should include the information that the lexical unit allows for passivization (attribute -diat, value pass).[6] Second, the grammatical rule (12) is formulated that makes it possible to derive the valency frame for passive usages of the verb, see (Kettnerová and Lopatková, 2009). On the basis of this rule, a derived valency frame for the verb *zvýhodnit*$_1$ 'to give an advantage' is generated[7] (see also Urešová and Pajas, 2009; Urešová, 2011):

(12) $\boxed{\text{ACT}_1\ \text{PAT}_4\ \text{MEANS}_7^{\text{typ}} \ \Rightarrow\ \text{ACT}_{7,\text{od}+2}\ \text{PAT}_1\ \text{MEANS}_7^{\text{typ}}}$

Let us focus on the examples introduced by Roland Wagner now. In general, the forms introduced by *jako* 'as' represent (as RW pointed) a tricky question in the description of the Czech language: *jako* – which can introduce both prepositionless nouns and prepositional noun groups (and clauses as well) – has an unclear morphological status and the case of the nominal varies depending on the syntactic context, as examples (6)–(8) demonstrate. The following example sheds more light on the problem of valency complementations that are introduced with the expression *jako* and the way it can be treated within the descriptive apparatus of FGD (and VALLEX in particular).

(13) *Občanský princip lidských práv chápal jako její základní prvek/hodnotu, nikoli vyčerpávající cíl a smysl.* (PDT, the word *hodnotu* 'value' added due to the morphological ambiguity of *prvek* 'component')
En. He viewed the civil principle of human rights as her substantial component/value, not as an overall aim and sense.

(14) *Občanský princip lidských práv byl chápán jako její základní prvek/hodnota, nikoli vyčerpávající cíl a smysl.* (PDT, modified)
En. The civil principle of human rights was viewed as its substantial component/value, not as an overall aim and sense.

(15) *chápat*$_2$ 'to interpret' … $\text{ACT}_1\ \text{PAT}_4\ \text{EFF}_{\text{jako}+4}$
-diat: pass, deagent, res-být

[5]The abbreviation 'typ' denotes so called 'typical' free modifications as they were introduced in VALLEX; they are typically related to some verbs (or even to whole classes of them) but they do not enter the core valency frame.

[6]Whereas the proposal of the structure of the VALLEX lexicon has been already published and discussed in the linguistic forum, an on-line version of the lexicon with explicit information on possible diatheses (and lexicalized alternations) is under development (a new lexicon release is planned at the end of 2015).

[7]Note that the instrumental form of the ACTor in a passive sentence is possible but it cannot be combined with an instrumental MEANS.

Further, the prepositional group *od*+Gen of ACT is rare in the corpus data but it is not excluded as the following example illustrates: *Obce, ve kterých se bude důsledně třídit sklo, jsou zvýhodněny při platbě odměn od společnosti EKO-KOM.* (from the Czech National Corpus (CNC), SYN series, https://kontext.korpus.cz/).

The verb *chápat*$_2$ 'to interpret' is characterized by the valency frame given in (15) for an unmarked active usage (as in (13)). The verb can be definitely used also in a passive construction, see (14). Then the passivization affects not only the form of the ACTor and PATient complementations, but also the form of the EFFect complementation (*jako*+Acc → *jako*+Nom) – all these changes are treated by the respective grammatical rule (16), which derives the valency frame for marked passive usages from the frame corresponding to the unmarked active ones provided in (15), see (Kettnerová and Lopatková, 2009):

(16) $\boxed{\mathsf{ACT}_1\ \mathsf{PAT}_4\ \mathsf{EFF}_{\mathsf{jako+4}} \ \Rightarrow\ \mathsf{ACT}_{7,\mathsf{od+2}}\ \mathsf{PAT}_1\ \mathsf{EFF}_{\mathsf{jako+1}}}$

3. FGD solution of the seeming collision

RW's examples represent a prototypical case of such syntactic operation as mentioned above. Let us illustrate the proposed cooperation of the grammar and lexical components of FGD in the description of this phenomenon.

In the VALLEX lexicon, possible reflexivization of the verbal participant coreferential with the subject is indicated by the presence of the value cor_k in the attribute reflexivity (-rfl; the index k encodes the morphemic case, i.e., 4 for accusative and 3 for dative). This value – identifying unambiguously the complementation that can be reflexivized[8] – is introduced for each lexical unit of a verb allowing for reflexivization of a particular member of a core valency frame (i.e., inner participants either obligatory or optional, and obligatory free modifications). For instance, in the lexical entry of the verb *obdivovat*$_1$ 'to admire', the attribute reflexivity records the information on the possibility of the accusative PATient to be reflexivized, see (17), and the verb usages in examples (18)–(19); whereas in (18) the slot for PATient is filled by *žáky* 'pupils', in (19) the reflexive *se* fills this slot (the coreferential items are marked by the index i in the examples).

[8] From the theoretical point of view, it would be more appropriate to specify reflexivity in terms of functors of valency complementations (not in terms of morphemic forms). However, the information on reflexivity is not complete in the VALLEX lexicon at present, see below. Thus we prefer to use special values (cor_3, cor_4) not to make an impression that all instances of possible reflexivization of individual valency members are recorded.

In the current version, reflexivity is captured only in such cases when a participant can be lexically expressed by the clitic forms of the reflexive pronoun *se/si* (certainly, also the non-clitic forms *sebe/sobě* may be used here due to the substitutability criterion (according to which the clitic forms can be substitute by the non-clitic forms if the occurrence of the reflexive stands for the pronoun)). However, VALLEX does not encode cases where the non-clitic variant of the reflexive pronoun is grammaticalized (i.e., prepositional groups, the instrumental and genitive case). The clitic variant has been given preference in the description of reflexivity due to the ambiguity of the clitic reflexives *se/si*, which produces severe problems for both human users and NLP tools.

(17) *obdivovat*$_1$ 'to admire' … ACT$_1$ PAT$_{4,\check{z}e,cont}$
-rfl: cor$_4$

(18) *Učitel obdivoval žáky, jak dobře zvládli výuku. (= žáci zvládli)*
En. The teacher admired pupils how well they managed the lessons.

(19) *Učitel se obdivoval, jak dobře zvládl neposlušné děti. (= sám sebe)*
the teacher$_i$ refl$_i^{clitic}$ admired how well (he$_i$) managed disobedient children
En. The teacher admired himself how well he managed disobedient children.

Let us return to RW's example of the verb *vnímat*$_3$ 'to see, to perceive'. Its valency frame in the meaning discussed here should have the following form in VALLEX:

(20) *vnímat*$_3$ 'to see, to perceive' … ACT$_1$ PAT$_{4,\check{z}e}$ EFF$_{jako+4}$
-rfl: cor$_4$

As the verb definitely allows for reflexivization of PATient, the attribute -rfl should provide the value cor4, see (20). As the morphemic form of EFFect is sensitive to syntactic context in which it is used – namely its form changes from *jako*+Acc into *jako*+Nom when the lexical unit is used in a reflexive construction with PATient lexically realized by the clitic form of the reflexive pronoun *se*, see (21) (RW correctly pointed out that the non-clitic long form of the reflexive pronoun *sebe* does not bring about such change, see (22)). The grammar component of FGD provides a formal syntactic rule capturing this change. This rule (as other rules describing changes in valency structure of verbs) allows for the derivation of the valency frame of the marked reflexive usage of the verb *vnímat*$_3$ 'to see, to perceive' (23) from the valency frame corresponding to an unmarked usage given in (20):

(21) *Otec se vnímá jako sok/jako génius.*
father$_i$ refl$_i^{clitic}$ sees as a rival/as a genius$_{i,Nom}$
(= otec se cítí někomu sokem/otec se pokládá za génia)
En. The father sees himself as a rival/as a genius.

(22) *Otec sebe (na rozdíl od matky) vnímá jako soka / * jako sok*
father$_i$ refl$_i^{non\text{-}clitic}$ (in constrast to the mother) sees as a rival$_{i,Acc}$ * as a rival$_{i,Nom}$
(svého syna).
(of his son)
En. (Contrary to the mother), the father sees himself as a rival (of his son).

(23) ACT$_1$ PAT$_{4,\check{z}e}$ EFF$_{jako+4}$ ⇒ ACT$_1$ PAT$_{4,\check{z}e}$ EFF$_{jako+1}$

The rule allowing for the generating the valency frame underlying the usage of a verb in reflexive constructions consists of a single change in the morphemic form of the EFFect complementation and its application is conditioned by the choice of the

clitic reflexive pronoun. The grammar module of FGD cooperates with the data stored in the lexical module where the possibility of the verb *vnímat*$_3$ 'to see, to perceive' to occur in reflexive constructions is specified in its lexical entry. On the same basis, the other verbs with EFFect changing its morphemic expression depending on the reflexive context (e.g., *deklarovat*$_2$ 'to declare', *hodnotit*$_1$ 'to evaluate', *chápat*$_2$ 'to perceive, to take as', *interpretovat*$_1$ 'to interpret', *nazývat/nazvat*$_1$ 'to call', *ohodnocovat/ohodnotit*$_1$ 'to rate', *označovat/označit*$_2$ 'to declare, to call', *pojímat/pojmout*$_3$ 'to comprehend, to conceive', *prezentovat*$_2$ 'to present', *přijímat/přijmout*$_{5,9}$ 'to accept', *stanovit*$_2$ 'to appoint', *určovat/určit*$_3$ 'to appoint, to designate', *ustavovat/ustavit*$_2$ 'to establish', *usvědčovat/usvědčit*$_2$ 'to convict', *uznávat/uznat*$_2$ 'to recognize', *vídat/vidět*$_5$ 'to see', *vnímat*$_3$ 'to see, to perceive', *vyhlašovat/vyhlásit*$_2$ 'to proclaim', *znát*$_1$ 'to know') indicated by RW as the source of "collision between two descriptive Principles of FGD" can be analyzed.[9]

4. Further grammatical aspects of the issue

We accept two issues from RW's study as most urgent for a further analysis: (i) The integration of the morphosyntactic change from *jako*+Acc into *jako*+Nom associated with the EFFect complementation into the descriptive apparatus of FGD (which we have addressed in Sections 2 and 3) and (ii) the explanation of the congruence: possible alternative *jako*+Nom within the verbal reflexivity with the form *jako*+Acc for the EFFect complementation (as an obligatory or optional valency member) is discussed in this Section.

4.1. EFFect and COMPLement verbal complementations

In addition to the valency complementation EFFect, the forms introduced by *jako* 'as' (either with the accusative case or with the nominative case) can function also as a free modification COMPLement. We can notice that the change of the morphemic expression from *jako*+Acc into *jako*+Nom may in fact reflect a change in the dependency structure (namely the type of the complementation and the target of a coreferential link) of the sentence, which brings about a semantic shift, see examples (24)–(26) and their dependency trees in Figures 1–4 (in the examples, subscripts display coreferences captured by arrows in the trees).

(24) *Klaus* *vnímá své soky* *jako hráč.* (RW 25a)
Klaus$_{i,Nom}$ takes his$_i$ rivals$_{j,Acc}$ as a sportsman$_{i,Nom}$
En. Klaus takes his rivals as a sportsman. (= Klaus is a sportsman)

[9] Note that this type of constructions concerns not only the above mentioned verbs with the EFFect but we can observe the same change in the morphemic form of the optional COMPLement free modification with, e.g., the verbs *definovat* 'to define', *charakterizovat* 'to characterize', *identifikovat* 'to identify', *kvalifikovat* 'to qualify', *poznávat/poznat* 'to get to know', *předkládat/předložit* 'to introduce, to propose', *představovat/představit* 'to introduce', *vyfotografovat* 'to take a photo', *zachovávat /zachovat* 'to keep', *zapisovat/zapsat* 'to record, to register'.

(25) *Jak vnímáte Prahu jako architekt?* (RW 25b)
how (you$_{i,Nom}$) take Prague$_{j,Acc}$ as an architect$_{i,Nom}$
En. What do you as an architect think of Prague? (= you are an architect)

(26) *Klaus vnímá své soky jako hráče.*
Klaus$_{i,Nom}$ takes his$_i$ rivals$_{j,Acc}$ as sportsmen$_{j,Acc}$
En. Klaus takes his rivals as sportsmen. (= Klaus's rivals are sportsmen)

(27) *Jako křesťan vnímám lidský život jako dar Boží,*
(I$_{i,Nom}$) as a Christian$_{i,Nom}$ take a human life$_{j,Acc}$ as God's gift$_{j,Acc}$
s nímž nemám právo nakládat.
which I have no right to treat
En. I as a Christian see a human life as a God's gift which I have no right to treat.

(28) *R. Steiner se jako tvůrce teosofie vždy chápal*
R. Steiner$_{i,Nom}$ refl$^{clitic}_{i,Acc}$ as an author$_{i,Nom}$ of theosophy always perceived
především jako okultista.
primarily as an occultist$_{i,Nom}$
En. R. Steiner, as an author of the theosophy, always perceived himself primarily as an occultist. (CNC, modified (*jako tvůrce teosofie* 'as an author of theosophy' added))

(29) *vnímat*$_4$ 'to see, to perceive' … ACT$_1$ PAT$_{4,že}$ MANN
-rfl: cor$_4$

The verb *vnímat* 'to see, to perceive' in (24) is described as the lexical unit *vnímat*$_4$ in VALLEX with obligatory MANNer, see (29) (as RW also suggests). Then the complementation expressed as *jako*+Nom has the function of an optional COMPLement (*Klaus, jsa hráč(em)* 'Klaus being a sportsman'), see Figure 1; the obligatory MANN is not present in the surface structure (it can be understood as *Klaus vnímá své soky způsobem, jak to dělají hráči* 'Klaus takes his rivals in the same manner as sportsmen do'). In sentence (25), the form *jako*+Nom clearly documents the function of COMPLement (*jakožto architekt* 'as being an architect'), with the pronominal adverb *jak* 'how' filling the MANNer valency position of *vnímat*$_4$, see Figure 2.

On the other hand, in example (26), *vnímat*$_3$ is used and the regular form for EFFect (*jako*+Acc) is used, see its valency frame (20); Figure 3 displays the dependency structure of the sentence.

An interesting example (27) with the verb *vnímat*$_3$ illustrates that the forms with *jako* 'as' can be used in both meanings in a single sentence: *jako*+Nom in *jako křesťan* 'as a Christian' has a function of COMPLement, whereas *jako*+Acc in *jako dar Boží* 'as God's gift' is EFFect (the substitution *jako nadílku Boží* 'as God's gift' – documenting the case form more transparently – may be used here), see see Figure 4.

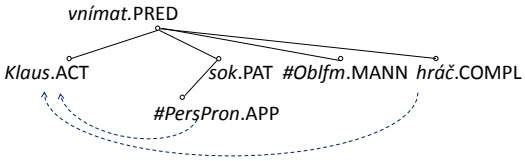


Figure 1. Dependency structure of sentence (24) Klaus vnímá své soky jako hráč.

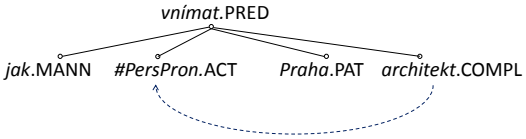


Figure 2. Dependency structure of sentence (25) Jak vnímáte Prahu jako architekt?

Moreover, example (28) (though rare in the corpus data) demonstrates that in case of the reflexive construction with the clitic variant of the reflexive pronoun both COMPLement and EFFect (if they are present) are expressed in nominative.

4.2. Agreement for **EFF**ect and **COMPL**ement complementations

Let us return to the issue of agreement for EFFect and COMPLement complementations in general. Based on the discussion presented below, we would like to clarify an appropriateness of different cases agreement in sentences (30)–(34).

(30) *Otec vnímá (svého) syna jako soka.*
the father$_j$ perceives (his$_j$) son$_{i,Acc}$ as a rival$_{i,Acc}$
En. The father perceives his son as a rival. (= son is a rival)

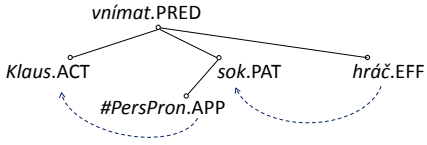


Figure 3. Dependency structure of sentence (26) Klaus vnímá své soky jako hráče.

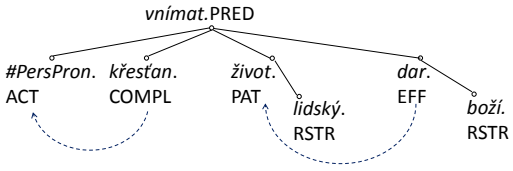


Figure 4. Dependency structure of sentence (27) Jako křesťan vnímám lidský život jako dar Boží, …

(31) *Otec* *se* *vnímá* *jako sok / jako génius.*
the father$_{i,Nom}$ reflclitic perceives as a rival$_{i,Nom}$ / as a genius$_{i,Nom}$
(= *otec se cítí někomu sokem/otec se pokládá za génia*)
En. The father perceives himself as a rival / as a genius.
(= father is a rival/genius)

(32) * *Otec* *se* *vnímá* *jako soka (svého syna).*
the father$_{i,Nom}$ refl$^{clitic}_{i}$ perceives * as a rival$_{i,Acc}$ (of his$_i$ son)

(33) *Otec* *sebe* *(na rozdíl od matky) vnímá*
the father$_{i,Nom}$ refl$^{non\text{-}clitic}_{i}$ (unlike the mother) perceives
jako soka (svého syna).
as a rival$_{i,Acc}$ (of his$_i$ son)
En. (Unlike the mother,) the father perceives himself as a rival (of his son).

(34) * *Otec* *sebe* *(na rozdíl od potomků) vnímá* *jako génius.*
the father$_{i,Nom}$ refl$^{non\text{-}clitic}_{i}$ (unlike children) perceives * as a genius$_{i,Nom}$

Both RW as well as the authors of this response do not accept the proposal given by Oliva (2000, 2001) according to which the form *se* plays the role of particle without its sentence function in all occurrences.[10] Then other arguments for the distinction between the pairs of examples (31)–(32) and (33)–(34) have to be found. Looking for such arguments, it turns up to be an analogy of the "mysterious" complement agree-

[10] According to Oliva's proposal, the following sentences (a)–(c) have (i) different lemmas (*vidět* for (a), (c) and *vidět se* for (b)) and (ii) different syntactic structures (transitive verb in (a), (c) and intransitive verb in (b)).

(a) *Vidím tě.* vs. (b) *Vidím se.* vs. (c) *Vidím sebe.*

Such analysis neglects parallelism in morphological paradigms of the non-reflexive and reflexive pronouns (as pointed out by Wagner, 2014) and suppresses syntactic parallelism of the structures with (almost) identical meaning structure. Moreover, Oliva's interpretation of all clitic reflexives as particles impedes the explanation of reciprocity. See esp. (Panevová, 2001; Komárek, 2001; Wagner, 2014).

As a result, the treatment of the reflexives proposed by Oliva would lead to large (and theoretically inadequate) expansion of the lexical data.

ment pointed out in the arguments of Oliva (2000) in favor of his proposal. The same arguments appeared also in the old observation made by Havránek (1928), see below.

The alternative description given by Panevová (2001, 2008) is based on the difference between possible antecedents (sources) for agreement in the case and number of an analyzed complementation. Her analysis can be exemplified on examples (35)–(38): In (37) and (38) there is only one source[11] of agreement, i.e. *chlapec* 'boy', while in (35) and (36) two possible sources of agreement (*matka* 'mother' and *chlapec* 'boy') are present. The choice of the source of agreement is semantically motivated: whereas in (35), it is *chlapec, který je umyt celý* 'the boy who is entire washed', and thus, it is *chlapce* 'boy$_{Acc}$' that is chosen as the source of agreement; in (36), it is *matka, která je celá uplakaná* 'mother who is entirely tearful' that represents this source. To summarize, examples (35) and (36) differ with respect to the sources for agreement and this difference is reflected in the change of the form of the target of agreement (Acc in (35), Nom in (36)).

The structure of sentence (37) is parallel to (35), the source of agreement remains the same, i.e., the reflexive pronominal complementation in accusative. The only change consists in the additional coreferential link between the reflexive pronoun *sebe* and the ACTor *chlapec* 'boy'. Although in example (38), the structure analogical to examples (35) and (37) is theoretically expected, the source of agreement differs – here it is not the complementation in the accusative case, but the nominative complementation.

(35) *Matka umyla chlapce celého.*
mother washed the boy$_{i,Acc}$ whole$_{i,Acc}$
En. The mother washed the entire boy. (= the boy was entirely washed)

(36) *Matka umyla chlapce celá uplakaná.*
mother$_{i,Nom}$ washed the boy whole$_{i,Nom}$ tearful$_{i,Nom}$
En. Being entirely tearful, the mother washed the boy. (= the mother was tearful)

(37) *Sebe chlapec umyl celého (ale sestru ne).*
refl$^{non\text{-}clitic}_{i,Acc}$ the boy$_{i,Nom}$ washed whole$_{i,Acc}$ (but not his$_i$ sister)
En. The boy washed himself entirely (but not his sister).

(38) *Chlapec se umyl celý.*
the boy$_{i,Nom}$ refl$^{clitic}_{i,Acc}$ washed whole$_{i,Nom}$
En. The boy washed himself entirely.

The tendency of the complement to agree as to the congruence with the subject in nominative when the clitic variant of the reflexive pronoun is present has been

[11] The terminology controller and target in the domain of congruence is used by Corbett (2006); he admits also the terms source or trigger (see Corbett, 2000, 2006). We prefer the term source here instead of the term controller (used within FGD for coreferential relations).

already reflected by Havránek (1928), see (39). According to the author, the accusative congruence – being rare already in the Old Czech – is limited to cases when the clitic reflexive pronoun does not in fact refer to the ACT or himself but to his (future or past) vision (thus a speaker sees himself as someone else). See Havránek's examples (40) and (41) interpreted by the author as acceptable in the context of memories (40) or in the situation when a speaker was making a double of himself (41).

(39) *cítí se zdráv* (Havránek)
(he$_{i,Nom}$) feels refl$^{clitic}_{i,Acc}$ fit$_{i,Nom}$
En. he feels fit

(40) *viděl se ležícího u řeky* (Havránek)
(he$_{i,Nom}$) saw refl$^{clitic}_{i,Acc}$ lying$_{i,Acc}$ by the river
En. he saw himself lying by the river

(41) *udělal se tlustýho* (Havránek)
(he$_{i,Nom}$) made refl$^{clitic}_{i,Acc}$ fat$_{i,Acc}$
En. he made himself fat

The corpus data support Havránek's interpretation also in the contemporary Czech, see examples (42) and (45) and their paraphrases (43) and (46), respectively, substantiating the semantic shift brought about by the accusative and nominative congruence. In (42) the speaker describes himself in the future: the speaker is not actually the man who has a house, a family and children at present but it is his future vision of himself. The paraphrase with the nominative congruence is much more suitable in the present context: in the situation when the speaker actually has a house, a family and children, see (43). In (45), the speaker disapprovingly characterizes the president of the Czech Republic Miloš Zeman; the accusative congruence emphasizes the speaker's disapproval: the president sees himself as a wise man but he is not actually wise.

According to our introspective, although the nominative congruence for expressing the same meanings as in (42) and (45), respectively, is not entirely excluded, the accusative agreement sounds more suitable for expressing that the PATient – despite being lexically realized by the reflexive pronoun – is not in fact referentially identical with the ACTor but it is rather a vision of himself, see (44) and (46). However, sparse corpus data do not allow us to make any definitive conclusions about the semantic shift between accusative and nominative congruence.

(42) *„Kdybych si měl představit sám sebe za deset let,* ***vidím se jako člověka,***
"if I should imagine myself in ten years, **(I$_i$) see refl$^{clitic}_{i,Acc}$ as a man$_{i,Acc}$**,
který má dům, rodinu a děti," *dodává.* (CNC)
who has a house, a family and children," he is adding
En. "If I should imagine myself in ten years, **I will see myself as a man** who has a house, a family and children," he is adding.

(43) ***Vidím se jako člověk,*** *který má dům, rodinu a děti,"*
(I$_i$) see refl$^{clitic}_{i,Acc}$ as a man$_{i,Nom}$, who has a house, a family and children,"
dodává. (CNC, modified)
he is adding.
En. "**I see myself as a man** who has a house, a family and children," he is adding.

(44) ? *„Kdybych si měl představit sám sebe za deset let,* ***vidím se jako člověk,***
"if I should imagine myself in ten years, **(I$_i$) see refl$^{clitic}_{i,Acc}$ as a man$_{i,Nom}$,**
který má dům, rodinu a děti," *dodává.* (CNC, modified)
who has a house, a family and children," he is adding.
En. "If I should imagine myself in ten years, **I will see myself as a man** who has a house, a family and children," he is adding.

(45) *Ale zároveň je miluje, protože zvětšují jeho důležitost,*
However, he$_i$ loves them magnifying his importance,
dávají mu gloriolu významné osobnosti, poskytují mu možnost v narcistním opojení
giving him VIP's glory, giving him the opportunity in a narcissistic intoxication
slyšet sama sebe, ***vidět se jako moudrého člověka,***
to hear himself, **to see refl$^{clitic}_{i,Acc}$ as a wise man$_{i,Acc}$,**
který nemá na politické scéně, ne-li mnohem dál,
who has not a rival on the political scene, if not even much further,
ani po tolika letech valnou konkurenci. (CNC)
after so many years
En. However, he loves them magnifying his importance, giving him VIP's glory and opportunity, in a narcissistic intoxication, to hear and **see himself as a wise man** who has not a rival on the political scene, if not even much further, after so many years.

(46) *Ale zároveň je miluje, protože zvětšují jeho důležitost,*
However, he$_i$ loves them magnifying his importance,
dávají mu gloriolu významné osobnosti, poskytují mu možnost v narcistním opojení
giving him VIP's glory, giving him the opportunity in a narcissistic intoxication
slyšet sama sebe, ***vidět se jako moudrý člověk,*** *[…]*
to hear himself, **to see refl$^{clitic}_{i,Acc}$ as a wise man$_{i,Nom}$,** […]
En. However, he loves them magnifying his importance, giving him VIP's glory and opportunity, in a narcissistic intoxication, to hear and **see himself as a wise man** […].

The nominative congruence – which is predominant in the reflexive constructions with the clitic form of the reflexive pronoun – has not yet been satisfactorily accounted for in the Czech linguistics. Karlík (1999) pointed out that the clitic variants of the

Czech personal pronouns generally exhibit morphosyntactic properties similar to affixes to a greater (the reflexive pronoun) or lesser (the non-reflexive pronoun) extent, see examples given by Karlík (2000). On the other hand, he avoids Oliva's extreme viewpoint of all clitic reflexives as particles stressing that the non-clitic and clitic forms of the reflexives should be interpreted not dichotomously (i.e., either as pronouns, or as particles), but gradually. Among other morphosyntactic properties attesting that the clitic variants of the reflexive pronoun behave similarly to affixes, see the coordination test (47)–(48) and impossibility of separate usages in (49); Karlík introduces the nominative congruence addressed in this paper as well, see (50)–(51).

(47) * *Holí* *se* *a* *Pavla.* (Karlík)
(he$_i$) shaves * refl$^{clitic}_{i,Acc}$ and Paul

(48) *Holí* *sebe* *a* *Pavla.* (Karlík)
(he$_i$) shaves refl$^{non\text{-}clitic}_{i,Acc}$ and Paul
En. He shaves himself as well as Paul.

(49) *Kohos* *holil?* * *Se.* / *Sebe.* (Karlík)
who$_{Acc}$ (you) shaved * refl$^{clitic}_{i,Acc}$ refl$^{non\text{-}clitic}_{i,Acc}$
En. Who did you shave? Myself.

(50) Petr *se* *umyl* *celý.* (Karlík)
Peter$_i$ refl$^{clitic}_{i,Acc}$ washed whole$_{i,Nom}$
En. Peter washed himself entirely.

(51) Petr *umyl* *sebe* *celého.* (Karlík)
Peter$_i$ washed refl$^{non\text{-}clitic}_{i,Acc}$ whole$_{i,Acc}$
En. Peter washed himself entirely.

We propose a hypothesis that the changes in the case forms of EFFect introduced by *jako* 'as' – combined (i) either with the accusative case in constructions with PATient lexically expressed by the non-clitic, see (33), or (ii) with the nominative case with the clitic variant of the reflexive pronoun, see (31) – may have the same basis as the changes in the complement congruence lying in specific morphosyntactic properties of the clitic forms of the reflexive pronoun, as illustrated by Havránek's and Karlík's examples. However, we leave this question open as confirming this hypothesis represents a tricky task as the available corpus data[12] are too sparse to study the distribution of *se* vs. *sebe* in the nominative and accusative form with the funcEFF or COMPL functions. The ideas proposed by RW about the role of these forms in the functional sentence perspective and the contrasts among the sentence members are promising for the future research.

[12]Syntactically annotated PDT is too small for such phenomena. Morphologically annotated CNC is large enough; however, it is not easy to formulate corpus queries identifying relevant concordances necessary for our research.

In conclusion, let us remark that in addition to the "mysterious'' agreement of the complement expressing the EFFect/COMPLement members in constructions with the reflexive pronoun *se/sebe* in accusative, similar changes in the source of agreement appear in constructions with the dative case of the reflexive pronouns *si/sobě*. As it goes beyond the scope of this paper, we only note that studying the congruence changes in constructions with the dative reflexive pronoun *si/sobě* would be fruitful too.

(52) *Jan sobě jako vítězi koupil nové kolo.*
John$_i$ refl$^{non\text{-}clitic}_{i,Dat}$ as a winner$_{i,Dat}$ bought a new bike
En. John bought a new bike to himself as to the winner.

(53) *Sobě Jan jako vítěz koupil nové kolo.*
(to) refl$^{non\text{-}clitic}_{i,Dat}$ John$_i$ as a winner$_{i,Nom}$ bought a new bike
En. John as a winner bought a new bike to himself.

(54) *Jan si jako vítěz koupil nové kolo.*
John$_i$ (to) refl$^{clitic}_{i,Dat}$ as a winner$_{i,Nom}$ bought a new bike
En. John as a winner bought a new bike to himself.

(55) * *Jan si jako vítězi koupil nové kolo.*
John$_i$ * (to) refl$^{clitic}_{i,Dat}$ as a winner$_{i,Dat}$ bought a new bike

5. Conclusion

Ronald Wagner´s critical remarks stimulated our deeper analysis of the marginal (see the Appendix) but theoretically important aspects of the operation of reflexivization and its requirements on modification of verbal valency frames undergoing this syntactic operation.

We have clarified here the criterion for delimitation of different lexical units within FGD (Principle 1) – when using the test of differences in valency frames, we restrict ourselves only to those changes that appear in active, nonreflexive, nonreciprocal constructions.

We have focused especially on the apparatus proposed in FGD (and the valency lexicons PDT-VALLEX and VALLEX elaborated within this theoretical framework) that allows for the effective description of paradigmatic changes in valency frames of Czech verbs related not only to grammatical diatheses but also to reciprocity; we have shown that it can be easily adopted for the description of reflexivity (as addressed by Ronald Wagner) as well.

Further, we propose a preliminary hypothesis on the alternation between *jako*+Acc and *jako*+Nom: some of them are semantically conditioned (EFFect vs. COMPLement), the other reflect the grammatical requirements (reflexivity). Since this analysis could not be based on extensive corpus data (due to the low frequency of the studied constructions in corpora, see also the Appendix), our conclusion is only preliminary and requires further research.

We have demonstrated that there can be observed a strong parallelism between accusative and nominative congruence of complements and the constructions with the reflexive pronoun, which indicates that the focused changes in congruence in reflexive constructions might have the same basis given by specific morphosyntactic status of the clitic forms of the reflexive pronoun. We have pointed out that it would be beneficial to extend the analysis to the dative reflexive pronoun *si* vs. *sobě*, which has not been focused in the syntactic description so far.

Acknowledgements

The research reported in this paper has been supported by the Czech Science Foundation GA ČR, grant No. P406/12/0557. This work has been using language resources developed and/or stored and/or distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013).

Bibliography

Cinková, Silvie. From PropBank to EngValLex: Adapting the PropBank-Lexicon to the Valency Theory of the Functional Generative Description. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 2170–2175, Genova, Italy, 2006. ELRA.

Corbett, Greville G. *Number*. Cambridge University Press, Cambridge, 2000.

Corbett, Greville G. *Agreement*. Cambridge University Press, Cambridge, 2006.

Hajič, Jan, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. Prague Dependency Treebank 2.0. LDC2006T01, Linguistic Data Consortium, Philadelphia, PA, USA, ISBN 1-58563-370-4, Jul 2006, 2006. URL `http://ufal.mff.cuni.cz/pdt2.0/`.

Havránek, Bohuslav. *Genera verbi v slovanských jazycích I.* Nová řada (VIII), čís. 2, Edice: Rozpravy Královské české společnosti nauk. Třída filosoficko-historicko-jazykozpytná. Kr. česká spol. nauk, Praha, 1928.

Karlík, Petr. Reflexiva v češtině. *Přednášky a besedy z XXXII. běhu Letní školy slovanských studií*, pages 44–52, 1999.

Karlík, Petr. Hypotéza modifikované valenční teorie. *Slovo a slovesnost*, 61(3):170–189, 2000.

Kettnerová, Václava and Markéta Lopatková. Changes in Valency Structure of Verbs: Grammar vs. Lexicon. In Levická, Jana and Radovan Garabík, editors, *Proceedings of Slovko 2009, NLP, Corpus Linguistics, Corpus Based Grammar Research*, pages 198–210, Bratislava, 2009. Slovenská akadémia vied.

Kettnerová, Václava and Markéta Lopatková. The Lexicographic Representation of Czech Diatheses: Rule Based Approach. In Majchráková, Daniela and Radovan Garabík, editors, *Natural Language Processing, Multilinguality*, pages 89–100, Bratislava, Slovakia, 2011. Slovenská akadémia vied, Tribun EU.

Kettnerová, Václava, Markéta Lopatková, and Eduard Bejček. The Syntax-Semantics Interface of Czech Verbs in the Valency Lexicon. In Fjeld, Ruth and Julie Torjusen, editors, *Proceedings of the 15th EURALEX International Congress*, pages 434–443, Oslo, Norway, 2012a. Department of Linguistics and Scandinavian Studies, University of Oslo.

Kettnerová, Václava, Markéta Lopatková, and Zdeňka Urešová. The Rule-Based Approach to Czech Grammaticalized Alternations. In Sojka, Petr, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 15th International Conference, TSD 2012. Proceedings*, number 7499 in LNCS, pages 158–165, Berlin / Heidelberg, 2012b. Springer Verlag.

Kolářová, Veronika. *Valence deverbativních substantiv v češtině (na materiálu substantiv s dativní valencí)*. Karolinum, Praha, 2010.

Komárek, Miroslav. Několik poznámek k reflexi reflexivity reflexiv. *Slovo a slovesnost*, 62:207–209, 2001.

Lopatková, Markéta, Zdeněk Žabokrtský, and Václava Kettnerová. *Valenční slovník českých sloves*. Nakladatelství Karolinum, Praha, 2008. (with co-authors: K. Skwarska, E. Bejček, K. Hrstková, M. Nová, M. Tichý).

Oliva, Karel. Hovory k „sobě/si/sebe/se“. In Karlík, Petr and Zdenka Hladká, editors, *Čeština – univerzália a specifika, Sborník konference ve Šlapanicích U Brna*, volume 2, pages 167–171, 2000.

Oliva, Karel. Reflexe reflexivity reflexiv. *Slovo a slovesnost*, 62:200–207, 2001.

Panevová, Jarmila. Problémy reflexivního zájmena v češtině. In *Sborník přednášek z 44. běhu Letní školy slovanských studií*, pages 81–88, Praha, 2001. UK FF.

Panevová, Jarmila. Problémy se slovanským reflexivem. *Slavia*, 77(1-3):153–163, 2008.

Panevová, Jarmila and Magda Ševčíková. The Role of Grammatical Constraints in Lexical Component in Functional Generative Description. In Apresjan, Valentina, Boris Iomdin, and Ekaterina Ageeva, editors, *Proceedings of the 6th International Conference on Meaning-Text Theory (MTT 2013)*, pages 134–143, Praha, 2013. Univerzita Karlova v Praze.

Sgall, Petr. *Generativní popis jazyka a česká deklinace*. Academia, Praha, 1967.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Reidel, Dordrecht, 1986.

Šindlerová, Jana and Ondřej Bojar. Towards English-Czech Parallel Valency Lexicon via Treebank Examples. In *Proceedings of 8th Treebanks and Linguistic Theories Workshop (TLT)*, pages 185–195, Milano, Italy, 2009. Università Cattolica del Sacro Cuore.

Urešová, Zdeňka and Petr Pajas. Diatheses in the Czech Valency Lexicon PDT-Vallex. In Levická, Jana and Radovan Garabík, editors, *Slovko 2009, NLP, Corpus Linguistics, Corpus Based Grammar Research*, pages 358–376, Bratislava, 2009. Slovenská akadémia vied.

Urešová, Zdeňka. *Valence sloves v Pražském závislostním korpusu*, volume 8 of *Studies in Computational and Theoretical Linguistics*. Institute of Formal and Applied Linguistics, Prague, 2011.

Wagner, Roland. A case of collision in principles of language description? *The Prague Bulletin of Mathematical Linguistics*, 101:123–146, 2014.

Appendix

The following Table 1 summarizes rough estimations of the frequency of the studied phenomenon in CNC –- the entire SYN (Synchronic written corpora) series was used.

Comment. The columns in Table 1 store the following information:

lemma …verb lemma (the slash mark separates imperfective and perfective lemmas (if applicable))

SYN …number of occurrences of the specified verb in the entire SYN series

(i) sample query … `[lemma="vnímat"]`

no VS …number of occurrences of the specified verb excluding the past participle/passive forms

(ii) sample query … `[lemma="vnímat" & tag="V[^s].*"]`

***jako*+Acc** …number of occurrences of the specified verb excluding the past participle/passive forms that co-occur with the word *jako* immediately followed by a wordform in accusative

(iii) positive filter on the results of query (ii):
– interval $[-5; 5]$ including KWIC
– positive filter `[word="jako"][tag="....4.*"]`

***jako*+Nom** …number of occurrences of the specified verb excluding the past participle/passive forms that co-occur with the word *jako* immediately followed by a wordform in nominative

(iv) positive filter on the results of query (ii):
– interval $[-5; 5]$ including KWIC
– positive filter `[word="jako"][tag="....1.*"]`

jako*+Nom with *se …number of occurrences of the specified verb excluding the past participle/passive forms that co-occur with the word *jako* immediately followed by a wordform in nominative and combined with the word *se*

(v) positive filter on the results of query (iv):
– interval $[-5; 5]$ including KWIC
– positive filter `[word="se"]`

ratio (%) …ratio of the result from (v) (i.e., occurrences of the specified verb excluding the past participle/passive forms that co-occur with the word *jako* immediately followed by a wordform in nominative and combined with the word *se*) related to the number of occurrences of the specified verb in the entire SYN series (column SYN)

lemma	SYN	no Vs	*jako*+Acc	*jako*+Nom[1]	*jako*+Nom with *se*[1]	ratio (%)
deklarovat	16 763	15 192	360	547	307	**1,83**
hodnotit	199 363	180 225	9 587	4 523	596	0,30
chápat	194 218	184 175	23 792	4 100	985	0,51
interpretovat	16 409	13 604	1 528	822	195	**1,19**
nazývat /	66 095	56 572	214	328	110	0,17
/ *nazvat*	63 299	58 744	347	496	146	0,23
ohodnocovat /	461	389	2	2	0	0,00
/ *ohodnotit*	17 445	12 764	774	364	20	0,11
označovat /	88 882	68 399	3 263	3 872	2 171	**2,45**
/ *označit*	218 550	186 784	7 050	3 159	606	0,28
pojímat /	4 802	3 780	1 074	477	67	**1,39**
/ *pojmout*	39 781	35 577	4 946	1 647	116	0,29
prezentovat	105 628	92 518	3 169	4 755	3 405	**3,22**
přijímat /	111 143	101 152	1 542	962	130	0,12
/ *přijmout*	351 964	298 849	4 684	2 668	182	0,05
stanovovat /	21 323	20 840	160	139	47	0,22
/ *stanovit*	184 129	118 425	2 342	853	182	0,10
určovat /	55 632	53 135	216	332	44	0,08
/ *určit*	272 499	112 202	2 536	429	57	0,02
ustavovat /	685	643	21	7	1	0,15
/ *ustavit*	14 868	8 647	90	168	100	0,67
usvědčovat /	3 799	3 697	18	12	2	0,05
/ *usvědčit*	14 064	10 564	54	33	3	0,02
uznávat /	48 989	45 816	1 512	891	132	0,27
/ *uznat*	123 000	108 597	1 904	1 099	70	0,06
vidat /	17 304	16 624	146	130	38	0,22
/ *vidět*	1 249 642	1 240 557	19 058	10 349	1 054	0,08
vnímat	148 961	131 121	26 378	8 932	982	0,66
vyhlašovat /	31 042	29 224	115	162	42	0,14
/ *vyhlásit*	213 251	137 996	756	597	43	0,02
znát	606 505	606 505	9 026	5 074	528	0,09

[1] Including occurrences with errors in disambiguation, complements, deagentive constructions etc.

Table 1. Rough estimations of the frequency of the studied phenomenon in CNC

Address for correspondence:
Markéta Lopatková
lopatkova@ufal.mff.cuni.cz
Malostranské nám. 25, Prague 1, 118 00, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 102 OCTOBER 2014

INSTRUCTIONS FOR AUTHORS

Manuscripts are welcome provided that they have not yet been published elsewhere and that they bring some interesting and new insights contributing to the broad field of computational linguistics in any of its aspects, or of linguistic theory. The submitted articles may be:

- long articles with completed, wide-impact research results both theoretical and practical, and/or new formalisms for linguistic analysis and their implementation and application on linguistic data sets, or
- short or long articles that are abstracts or extracts of Master's and PhD thesis, with the most interesting and/or promising results described. Also
- short or long articles looking forward that base their views on proper and deep analysis of the current situation in various subjects within the field are invited, as well as
- short articles about current advanced research of both theoretical and applied nature, with very specific (and perhaps narrow, but well-defined) target goal in all areas of language and speech processing, to give the opportunity to junior researchers to publish as soon as possible;
- short articles that contain contraversing, polemic or otherwise unusual views, supported by some experimental evidence but not necessarily evaluated in the usual sense are also welcome.

The recommended length of long article is 12–30 pages and of short paper is 6–15 pages.

The copyright of papers accepted for publication remains with the author. The editors reserve the right to make editorial revisions but these revisions and changes have to be approved by the author(s). Book reviews and short book notices are also appreciated.

The manuscripts are reviewed by 2 independent reviewers, at least one of them being a member of the international Editorial Board.

Authors receive a printed copy of the relevant issue of the PBML together with the original pdf files.

The guidelines for the technical shape of the contributions are found on the web site `http://ufal.mff.cuni.cz/pbml`. If there are any technical problems, please contact the editorial staff at `pbml@ufal.mff.cuni.cz`.