The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014

EDITORIAL BOARD

Editor-in-Chief

Jan Hajič

Editorial staff

Martin Popel
Ondřej Bojar

Editorial Assistant

Kateřina Stuparičová

Editorial board

Nicoletta Calzolari, Pisa
Walther von Hahn, Hamburg
Jan Hajič, Prague
Eva Hajičová, Prague
Erhard Hinrichs, Tübingen
Aravind Joshi, Philadelphia
Philipp Koehn, Edinburgh
Jaroslav Peregrin, Prague
Patrice Pognan, Paris
Alexandr Rosen, Prague
Petr Sgall, Prague
Hans Uszkoreit, Saarbrücken

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

© 2014 PBML. All rights reserved.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014

CONTENTS

© 2014 PBML. All rights reserved.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014

EDITORIAL

With this issue, The Prague Bulletin of Mathematical Linguistics enters the second half of the century of its life. As we have summarized in the 100th issue published at the end of the last year, the first fifty years were full of events that were rather difficult to live with, but I am very happy that the journal has survived, and, if I may rather immodestly claim, without a loss of scientific prestige.

As a member of the group of computational linguists at Charles University, who were responsible for the founding of the journal, I had the opportunity to follow its development from the very beginning, becoming a member of the Editorial Board in 1969 and the Editor-in-Chief in 1977. I am convinced that it is time to pass over this role to younger generation and therefore I have decided to resign, staying on the Board just as one of its members.

At this occasion, I would like to thank most sincerely all the Board members for the continuous and very effective support without which we could not have survived and could not follow up the new trends in the contents and methodology of computational linguistics, to the authors who have contributed to broaden the range of the topics of the journal, and, last but not least, to the readers who have kept to follow the contents of the journal – and, eventually, cite it in their writings, which will help keep PBML on the lists of important publications.

I cordially wish the new Editor-in-Chief a full success and I am looking forward to our cooperation on the Board in the future.

Eva Hajičová

hajicova@ufal.mff.cuni.cz

© 2014 PBML. All rights reserved.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 7–28

Dynamic Models in Moses for Online Adaptation

Nicola Bertoldi

Fondazione Bruno Kessler

Abstract

A very hot issue for research and industry is how to effectively integrate machine translation (MT) within computer assisted translation (CAT) software. This paper focuses on this issue, and more generally how to dynamically adapt phrase-based statistical machine translation (SMT) by exploiting external knowledge, like the post-editions from professional translators.

We present an enhancement of the Moses SMT toolkit dynamically adaptable to external information, which becomes available during the translation process, and which can depend on the previously translated text. We have equipped Moses with two new elements: a new phrase table implementation and a new LM-like feature. Both the phrase table and the LM-like feature can be dynamically modified by adding and removing entries and re-scoring them according to a time-decaying scoring function. The final goal of these two dynamically adaptable features is twofold: to create additional translation alternatives and to reward those which are composed of entries previously inserted therein.

The implemented dynamic system is highly configurable, flexible and applicable to many tasks, like for instance online MT adaptation, interactive MT, and context-aware MT. When exploited in a real-world CAT scenario where online adaptation is applied to repetitive texts, it has proven itself very effective in improving translation quality and reducing post-editing effort.

1. Introduction

One of the hot topic in Machine Translation (MT) research is the integration of MT technology into Computer Assisted Translation (CAT) tools, and more specifically the efficient exploitation of human translator feedback to improve the MT system.

The ultimate goal of incorporating an MT system in a CAT tool is to increase the productivity of a human translator by providing her/him translation suggestions, which are as accurate as possible, as close as possible to her/his preferences, and as

© 2014 PBML. All rights reserved.
Corresponding author: bertoldi@fbk.eu
Cite as: Nicola Bertoldi. Dynamic Models in Moses for Online Adaptation. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 7–28. doi: 10.2478/pralin-2014-0001.

coherent as possible throughout a whole document from a lexical, semantic and syntactic perspective. To this purpose, the continuous exploitation of the post-edits of a human translator is crucial for updating the MT system embedded in the CAT tool and improving its quality. The above-mentioned task is a special case of the more general topic of adapting an MT system using dynamic techniques, i.e. the scenario in which a MT system is applied to a stream of input text, while simultaneously receiving a stream of informative data to improve the behavior of the system over time For instance, MT technology to translate a stream of news content could benefit from the external knowledge about the current news domain, if properly fed into the MT models.

The current standard MT technology is not completely appropriate to this online adaptation scenario, due to some intrinsic shortcomings. For efficiency purposes, translation is performed independently sentence by sentence, and the context of the previous and next sentences is not taken into consideration; hence, the lexical, syntactic and semantic consistency cannot be assured at the document-level. Most MT models are estimated on a predefined corpus in the preliminary training phase, and they cannot be modified during the translation process; if any change is required the MT system needs to reload the models, which is an expensive operation due to their potential huge size. MT systems do not have learning capabilities either, and hence they cannot be adapted and become responsive to translator feedback.

In order to overcome some of these limitations, MT technology should employ modifiable models. Here is a brief and partial list of desiderata for such dynamic models: the insertion and deletion of entries and the modification of of their scores should be permitted; the changes inside the models should be immediately available to the decoder without reloading the models; according to the task, the changes could be either local to the current text to translate or persistent over time.

A recent implementation of MT models, discussed later in the paper, relying on the dynamic suffix array data structure does not suffer from the limitations of the static models, but still it does do not satisfy all requirements.

This paper presents an enhanced version of the well-known and world-wide used Moses SMT toolkit (Koehn et al., 2007), providing simple, modular and affordable solutions to make it dynamic. New implementation types for the phrase table and the language model are created, which give the required functions, and additionally can be modified in any time by means of command line instructions.

The paper is organized as follows. Section 2 overviews the phrase-based Moses system. In Section 3 a comprehensive description of our proposed enhanced dynamic version of Moses is given; more specifically, details are supplied about the employed data structures, the scoring of the entries in the translation and language models, the way of communicating to the decoder and updating the models. Section 4 introduces the standard generic framework where the proposed dynamic system can be applied. Section 5 compares our approach to tackle the online MT adaptation to other solutions at disposal in Moses or described in the literature. Section 6 reports on the ef-

ficiency and effectiveness of the proposed solution. Finally, Section 7 ends the paper by presenting some ideas to further improve the dynamic models, some of which are currently under development.

2. Description of the Static System

A high level description of the Moses SMT toolkit is provided aiming at highlighting the elements we modified to build its dynamic enhanced version as well as the decoding process. Among the many variants of Moses the phrase-based decoder is considered.

2.1. Data Structure

Phrase-based Moses relies on one or more *phrase tables*, which provides translation alternatives, and on several *feature functions*, which provide the partial scores for the overall computation of the final translation score.

The phrase table is essentially a set of *translation options*. A translation option pairs a source phrase with one of its target translations and assigns a series of scores from the feature functions. The phrase table is looked up for getting all translation options associated to a given source phrase. In the current version of Moses, several types of phrase tables are implemented; the two most used types are based on prefix trees and suffix arrays. Both data structures are very efficient to store a huge amount of entries and to reply to a search request as fast as possible. For efficiency, the scores of the phrase-based translation model are stored in the same table.

Feature functions can be classified into three groups according to the information they depend on: (i) those based on both source and target words (phrase pairs); (ii) those relying on monolingual –usually target– words (n-grams); and (iii) those depending only on the size of the output. The translation model and the lexicalized reordering model clearly belong to the first group, and both use the data structure of the phrase table. The target language model, which belongs to the second group, is usually provided by a third-party toolkit, like SRILM (Stolcke, 2002), IRSTLM (Federico et al., 2008), randLM (Talbot and Osborne, 2007), kenLM (Heafield et al., 2013); however, a basic LM implementation is available in Moses as well. The feature functions belonging to the third group, like word and phrase penalty, do not require lexical information, but rather they are mostly evaluated on-the-fly during translation.

2.2. Translation Process

The translation of an input sentence is performed into two steps.

In the first phase, called *pre-fetching*, the translation options for all possible *spans*[1] of the input sentence are collected from the phrase table(s). In this phase, pruning of

[1] A *span* is a sequence of consecutive words of the input sentence.

the options is also performed in order to keep the translation process computationally manageable. Options are first ranked according to their score in the phrase tables and an estimate of their expected utility within the final translation, and then the k-best [2] options are retained, while the others are discarded.

In the second phase, called *decoding*, a possibly large number of translation hypotheses are built by (i) selecting a span segmentation of the input sentence, (ii) reshuffling the spans, and (iii) concatenating all translation options of all spans gathered in the pre-fetching phase with the constraint that all words on the input sentence are covered exactly once. Each translation hypothesis is scored by log-linearly interpolating all feature functions.

The heavy computational effort of the decoding stage is made affordable by exploiting very efficient algorithms, like beam search, multi-stack, or cube-pruning, constraints to limit the word reordering and the translation options, and data structures to store the huge amount of translation alternatives.

3. Description of the Dynamic System

The data structures provided by Moses for the phrase table and the lexicalized reordering table, and by other software for the language model are optimized for huge amounts of data and for quick access. Unfortunately, these structures are static and do not admit any change once they have been built and loaded. The suffix-array implementation of the phrase and lexicalized reordering table could be easily adapted for dynamic modification. A comparison between this implementation and ours is given in Section 5.

In order to make Moses dynamically adaptable we propose a new type of implementation for the language model and for the phrase table which are illustrated in Sections 3.1 and 3.2, respectively. The dynamic language model and phrase table behave similarly to the standard language model and phrase table implementations with the additional feature that the set of its accumulated entries (either phrase pairs or n-grams) and their associated scores can change over time, by inserting or re-scoring entries. Furthermore, each entry is associated with an *age*, which is the time the entry was inserted in the dynamic language model and phrase table, and its scores are made dependent on it; a rationale for that is given in Section 3.3.

The dynamic models live in memory and currently the scores of each entry cannot be saved on disk; hence their content is lost in case of exit. Nevertheless, they can be easily re-created by collecting all entries inserted until the exit with the correct age, for instance from an activity log, and by loading them when Moses restarts and the models are initialized. The informing data to update the models are communicated to the system adding a simple xml-based annotation in the input text. Appendix A provides details about the annotation.

[2] k is set by means of the parameter "`-ttable-limit`".

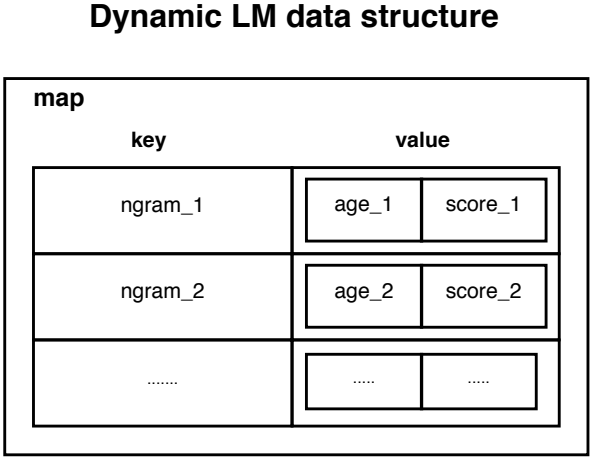

Figure 1. The data structure implementing the dynamic language. It relies on the map object provided by C++ Standard Template Library.

The dynamic system can be hence implemented by exploiting the dynamic phrase table and/or the dynamic language model in addition to or in substitution of the static models. Any method for combining the static and the dynamic phrase tables available in Moses can be ideally applied, namely fill-up, backoff, log-linear interpolation, etc. The dynamic language model must be used as an additional feature in Moses. No modification to the decoding phase is however required.

The source code of the proposed dynamic language model and phrase table is available in the branch "`dynamic-models`" under the official GitHub repository of Moses toolkit, directly accessible from this URL:
`https://github.com/moses-smt/mosesdecoder/tree/dynamic-models`

3.1. Dynamic Language Model

The dynamic language model, shown in Figure 1, is implemented by means of a simple data structure based on a C++ Standard Template Library[3] (STL) map. The map links a n-gram to its age and its age-dependent score. No restriction to the length of n-grams is given.

By means of this map the actions expected for the dynamic language model can be quickly performed: (i) inserting a new n-gram pair with an associated age; (ii) getting and updating the score of a given n-gram; (iii) deleting a specific n-gram; (iv) cleaning the dynamic language model.

[3] www.sgi.com/tech/stl

To insert an n-gram together with its age, first the age-dependent score is computed as described in Section 3.3, and then the data structure is updated. If the n-gram is new, it is added to the map with its age and score, otherwise its age and score are just updated. It is worth noting that the insertion of an n-gram, either existing or new, always implies the aging of all existing entries, and hence their re-scoring. The access and deletion as well the insertion of the n-grams are performed by means of the standard STL map functions. The computational cost is $\mathcal{O}(N)$ for insertion and $\mathcal{O}(\log(N))$ for access and deletion, where N is the total amount of stored n-grams. Insertion is more expensive because all existing n-grams must be aged.

Two modalities of lookup into the dynamic language model are implemented when Moses queries for the score of a target n-gram $(w_1, \ldots, w_l)$. In the first case (*AllSubStrings*), all its substrings of any length $(w_i, \ldots, w_j)$ are searched, their scores are averaged according to the following formula, which takes into account the number of substrings of a specific length, and the resulting score $avg\_score(w_1, \ldots, w_l)$ is actually returned.

$$avg\_score(w_1, \ldots, w_l) = \sum_{x=1}^{l} \frac{1}{x} \left(\sum_{i=1}^{l-x+1} score(w_i, \ldots, w_{i+x-1}) \right)$$

In the second case (*WholeString*) the whole string $(w_1, \ldots, w_l)$ is searched in the map receiving the stored $score(w_1, \ldots, w_l)$ is returned. The first modality exploits the dynamic language model as a simple lexicalized rewarding/penalty feature; instead, the second modality makes the dynamic language model more similar to a standard language model, since it simulates the summation of as many scores as the length of the n-grams. However, the dynamic language model is currently implemented as a stateless feature, and hence it does not support the computation of scores for n-grams across different translation options. This implementation choice is mainly justified by an efficiency reason: the lookup in the dynamic language model is performed only once and only for the n-grams included in the pre-fetched translation options; if we admitted the lookup of all possible n-grams created at translation time, like for a standard LM feature, the computational cost could become unaffordable.

3.2. Dynamic Phrase Table

The dynamic phrase table, which stores translation options and their associated ages and scores, is implemented by means of a two-tier data structure of maps and vectors, as sketched in Figure 2. Map and vector objects are supplied by the C++ STL. No restriction to the length of source and target phrases is given. The dynamic phrase table extends the Moses basic object *PhraseDictionary*.

The keys of the map consist of source phrases, and the corresponding values are pairs of pointers. The first pointer links to the current collection of translation options of the source phrase; for them, a specific data structure (*TranslationOptionCollection*)

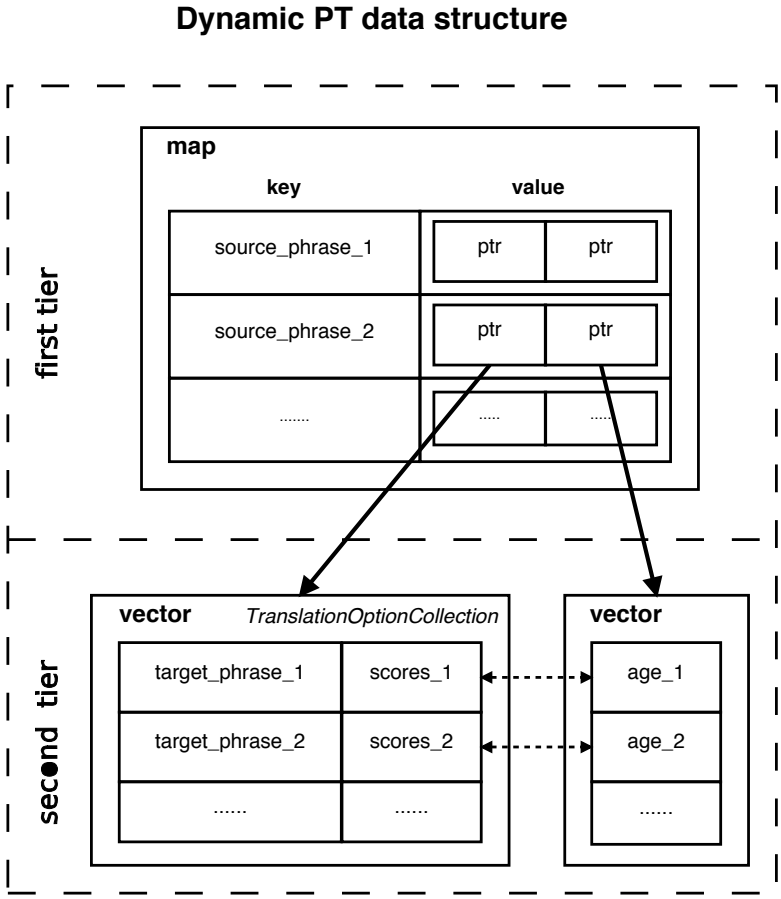

Figure 2. The two-tier data structure implementing the dynamic phrase table. It relies on map and vector objects provided by C++ Standard Template Library, and on the vector-based TranslationOptionCollection *object provided by Moses. Dashed lines are not real links; they just show that the entries of the two vectors are kept aligned.*

already available in Moses is exploited, which is essentially a vector of target phrases and associated scores. The score for each option is actually a vector of floating values. The second pointer links to a vector storing the ages associated to each translation options. The two vectors of target phrases and ages are kept aligned to simplify the access, modification, insertion and deletion of translation options.

The proposed data structure allows to perform the actions expected for the dynamic phrase table: (i) inserting a new phrase pair with an associated age; (ii) getting and updating the scores of a given phrase pair; (iii) returning the collection of translation options for a given source phrase; (iv) removing a specific phrase pair; (v) removing all translation options associated to a given source phrase; (vi) fully cleaning the dynamic phrase table.

To insert a phrase pair together with its age, first the new age-dependent score is computed as described in Section 3.3, and then the data structure is updated. Three cases are possible: (i) if a the source phrase does not exists a new entry in the map is inserted pointing to two newly created vectors, where the new target phrase and its age are inserted; (ii) if the source phrase exists but the target phrase does not, the new target phrase and its age are inserted just at the end of the two associated vectors, hence, keeping their alignment; (iii) if both source and target phrases exist, the corresponding entries in the two vectors are updated according to the new age and scores. It is worth noting that the insertion of a phrase pair, either existing or new, implies the aging of all existing entries, and hence their re-scoring.

To delete a phrase pair, first it is searched in the data structure, then the target phrase and the corresponding age are removed from both second-tier vectors, and finally the source phrase is removed from the first-tier map, if its corresponding translation option collection becomes empty. To delete all translation options associated to a given source phrase, first the two vectors linked to it are destroyed and then the source phrase is removed from the first-tier map. The cleaning of the whole dynamic model is done by recursively removing all source phrases.

To access the whole collection of the translation options of a specific source phrase, a quick lookup into the first-tier map is sufficient. The more expensive access to a specific phrase pair happens only when a new entry is added, but not during the pre-fetching and decoding phases.

The access to the whole collection of translation options for a given source phrase is performed very frequently in the pre-fetching phase of the decoding; hence, the data structure is optimized for minimizing the complexity of this action. More specifically, assuming that the dynamic phrase table stores S source phrases and T translation options for each source phrase on the average, and considered the complexity of the STL map- and vector-based functions employed, the global cost for inserting a phrase pair is $\mathcal{O}(T\,S)$, for deleting it is $\mathcal{O}(T\,\log(S))$, and for accessing the whole collection of translation options associated to a given source phrase is only $\mathcal{O}(\log(S))$. The insertion operation is expensive because all existing entries must be aged.

By default, the dynamic phrase table provides a vector of one score for each entry. However, it is possible to associate a vector of N values, by equally dividing the age-dependent score; this could be useful for instance if the dynamic phrase table is combined with a static phrase table, which has 4 scores by default.

3.3. Time-decaying Scoring Functions

The dynamic models presented in this paper aim at updating the SMT system by benefitting from information which becomes available during the translation of a document in an online framework. Such information can change over time according to the new text to translate, to the translations produced so far, and to the feedback received by an external knowledge source, like human post-editors and evaluators.

It is reasonable to give more importance to more recent update requests. Henceforth, we developed an approach, where each entry in the dynamic models is always associated with its age, which counts how many insertion operations have occurred after it was actually inserted. A score depending on such age is then associated to each entry according to the core policy that "the more recent, the worthier". This age-dependent score is actually exploited by the decoder during translation.

	hyperbola	power	exponential	cosine
$reward(age)$	$\frac{1}{age}$	$\sqrt[4]{\frac{1}{age}}$	$\exp\left(\frac{1}{age}-1\right)$	$\cos\left(\frac{\pi}{2}\frac{age-1}{MaxAge}\right)$
$penalty(age)$	$\frac{1}{age}-1$	$\sqrt[4]{\frac{1}{age}}-1$	$\exp\left(\frac{1}{age}-1\right)-1$	$\cos\left(\frac{\pi}{2}\frac{age-1}{MaxAge}\right)-1$

Table 1. Several types of reward and penalty scoring functions depending on the age of an entry.

Several scoring functions are defined as reported in Table 1. These functions operate like a decreasing reward, ranging from 1.0 to 0.0, or like an increasing penalty ranging from 0.0 to -1.0. During translation the Moses decoder could look for an entry which is not included in the dynamic model either because it was never inserted or because it became too old and hence it was removed. In this case, it receives a lower-bound no-match score fixed to 0.0 and to $penalty(MaxAge)$ for the reward and penalty scoring functions, respectively.

Distinct scoring functions and distinct values of $MaxAge$ can be configured for the dynamic phrase table and language model. Details how to configure the dynamic models are given in Appendix B.

4. Translation with a Dynamic SMT System

As mentioned in Section 1, a typical use of the proposed dynamic models is their exploitation in a SMT system, which adapts as soon as external (supervised) feedback is available, like in a CAT scenario.

1. initialize dynamic models M_d from files
2. estimate combined models M_c
3. initialize Suggestion Manager
4. for source sentence src_i, $i = 1 \ldots, N$
5. get annotation ann_i from Suggestion Manager
6. update M_d exploiting ann_i
7. update M_c
8. get translation tr_i of src_i exploiting M_c
9. (optionally) get post-edit pe_i of tr_i
10. update Suggestion Manager exploiting src_i, tr_i and pe_i

Figure 3. General procedure to translate with a dynamic SMT system. The informing data to update the dynamic models are generated by a Suggestion Manager, which optionally exploits human post-edits.

A static SMT system can translate segments of an input text in any order, and likely in parallel to save time; instead, a dynamic SMT system must proceed sequentially, and communicate synchronously with an external source to get the required feedback. The general procedure is sketched in Figure 3.

Let us assume to translate a document $D = \{src_1, \ldots, src_N\}$ composed of N segments. In the initialization phase, the dynamic models M_d are created by optionally loading entries from files (step 1), and the combined models M_c are estimated exploiting both M_d and standard static models (step 2).

An external module (*Suggestion Manager*) is supposed to be available, which provides the informing data to update the dynamic language model and phrase table in the format shown in Figures 2 and 3 in Appendix A. The Suggestion Manager potentially relies on the full input document, and on the portion of the translation produced so far. Hence, at the beginning it is initialized exploiting only D (step 3).

The translation of D proceeds sequentially sentence after sentence (step 4) as follows: first the informing data ann_i are requested from the Suggestion Manager (step 5); the dynamic models M_d are updated by exploiting ann_i (step 6) and combined into M_c (step 7); then, Moses produces the translation tr_i of the source sentence src_i exploiting the current combined model (step 8); optionally, the post-edit pe_i is collected (step 9), if human intervention is considered; finally, the Suggestion Manager is updated by exploiting src_i, tr_i, and pe_i (step 10).

It is worth noting that one instance of Moses runs continuously from the beginning, and it is not restarted for each input sentence; to achieve this goal Moses should be run in a client/server modality, no matter on which communication channel (standard I/O, socket, or other) it relies.

The dynamic system is efficient in an online MT adaptation task, if creating the informing data (step 5) and updating the combined model M_c and the Suggestion

Manager (steps 7 and 10, respectively) are fast enough. The update of M_d (step 6) is efficient per se thanks to our proposed implementation.

The log-linear interpolation provided by Moses is a good choice for an high-speed combination of static and dynamic models; in this case, the dynamic models are used as additional features which are configured as described in Appendix B. Combining dynamic and static phrase tables by means of fill-up, backoff, or linear interpolation, is also possible; however, in this case the dynamic table must be configured to store the same amount of scores as the static table. In principle, the combined model could be composed by the dynamic model only, and in this case no a-priori background knowledge is used. Vice-versa, the dynamic model could be completely disregarded, and in this case the system loses his adaptation capability.

Concerning the effectiveness of the proposed dynamic models, it is important to stress that the translation performance of the dynamic SMT system strongly depends on the quality of the informing data (phrase pairs and n-grams) generated by the Suggestion Module. Although the development and the description of a high-performing Suggestion Manager is out of the scope of the paper, it is worth mentioning main desiderata: (i) the informing data returned by the Manager consist of a possibly empty set of phrase pairs and n-grams from which the annotations are built; (ii) the informing data must depend on only the source segments, translations and post-edits available so far, but not necessarily on all of them; (iii) computation of the informing data must be reasonably fast.

A basic Python-based software `"onlineSMT.py"` was written implementing the online translation procedure described in Figure 3 and based on the dynamic SMT system; it is available in the directory `"scripts/online"` of the Github Moses branch `"dynamic-models"` (see Section 3 for its URL). With respect to the standard version of Moses, this software enables the sequential translation of the source sentences, the collection of the informing data from the Suggestion Manager, and the update of the dynamic models. Two Suggestion Managers were developed which provide the informing data (phrase pairs or n-grams): one relies on the Constrained Search algorithm (Cettolo et al., 2010); the other on a modified online version of mgiza++, and Moses phrase extractor. A third basic Suggestion Manager which does not create any informing data was also developed to mimic a static system within an online framework. Moreover, the dynamic wrapper optionally creates annotations not only for the dynamic models but also for the hard-coded ”xml-input” function of Moses. Usage instructions are available.

The software for tuning the Moses by means of Minimum Error Rate training procedure (Och, 2003) was also slightly modified, in order to properly handle the dynamic system; the new version `"mert_dynamic_moses.py"`, available in the same location, essentially substitutes the batch translation by means of the standard Moses with the online translation by means of the dynamic wrapper `"onlineSMT.py"`. The software allows the optimization of the additional weights for the dynamic models in a real online framework.

5. Comparison of Alternative Approaches for Online MT Adaptation

Moses includes several implementations of the phrase table, like OnDisk, Binary, Compress.[4] All of them aim at optimizing either memory consumption, access time, or disk usage; nevertheless they are static and cannot be modified once trained; hence they are not suitable in an online adaptation framework.

Moses is already able to modify its behavior at run-time by means of the "xml-input" function. The decoder can be fed with new phrase pairs and scores, which are used as exclusive or additional options for the sake of the translation. A few shortcomings of this approach are briefly reported: (i) the suggested options refer to a specific input span; (ii) it is not possible to provide options for overlapping spans; (iii) the suggested options are at disposal only for the current sentence, and then discarded; (iv) the LM is not modified at all.

Moses includes an implementation of the phrase table based on a suffix-array data structure. The phrase table is not created and scores are not computed in the training phase; the translation options are instead collected and scored on-the-fly at translation time, by means of an extremely efficient method of storing and searching the training corpus; word-alignment information for all sentence pairs should be also included in the training data. The suffix-array phrase table can be made suitable for online adaptation by extending the training corpus itself with the suggested options. Levenberg et al. (2010) presented stream-based translation models, based on a dynamic suffix array which allow to add and delete parallel sentences, and maintain the memory space bounds. To me, this phrase table implementation has a few weaknesses: (i) suggested options are merged into the whole training corpus; hence, it is not trivial rewarding them with respect to the others; (ii) the changes are persistent over time, because the new informing data are essentially fused into the training data; (iii) still, the LM is not modified at all.

The research about language model adaptation, conducted not only in MT but also in the speech processing area (Bellegarda, 2004), does not fit very well the online scenario considered here. In fact, to our knowledge most approaches to LM adaptation aim at re-estimating and/or re-tuning the internal LM parameters (i.e. n-gram probabilities) when new data appear, but they usually do not allow adding or deleting new n-grams on-the-fly. Levenberg and Osborne (2009) investigated in this direction; they presented an enhancement of the randomized language model, which is adapted by means of an efficient (in space and time) incremental training algorithm as soon as a small bunch of data becomes available, and which performs comparably to the corresponding batch re-trained LM. In our opinion, their approach, although very efficient, becomes infeasible in our online scenario, because the frequency of the adaptation step is extremely high, one every new sentence. Moreover, the deletion

[4] See the official Moses documentation for details and credits to authors.

of n-gram in the model is not controlled by external feedback, but just regulated by memory bounds.

Our new dynamic implementation of language model and phrase table overcomes the drawbacks of the mentioned approaches. In particular, (i) the entries inserted in the dynamic models are persistently available for the translation of the future sentences; (ii) however, they can be removed from the models at any time; (iii) if the suggested options refer to overlapping spans, the choice of the best alternative is made in the decoding phase by avoiding any potentially dangerous greedy decision taken before; (iv) due to the time-decaying scoring function, it is possible to reward or penalize specific translation options; (v) a way to dynamically update a LM-like feature is provided.

Other works proposed solutions for dynamic adaptation of MT systems, but either they did not make the source code publicly available, or they imposed stricter constraints than ours. For instance, Nepveu et al. (2004) described a dynamic system built at the word-level only and relying on IBM Model 2 to get word alignment. The dynamic systems based on caches proposed by Tiedemann (2010) and Gong et al. (2011) are updated "with the translation options used in the best (final) translation hypothesis of previous sentences", and hence they cannot embed a stream of external knowledge.

We have been certainly inspired from their ideas to implement the dynamic system, but we have created a more flexible and extendible infrastructure, making it available for further improvements.

6. Performance

The dynamic MT system presented in this work has already been successfully and effectively employed in an online MT adaptation scenario (Wäschle et al., 2013; Bertoldi et al., 2013). In these experimental investigations, we tested a variety of methods to generate informing data from the human post-edits, as well as several combinations of the dynamic models. We tried various settings on translation tasks in domains such as English-Italian Information Technology and English-German Patents. Relative improvements in terms of BLEU score up to 10% are reported over a state-of-the-art static system.

Here a brief summary of their analytic outcomes is reported.

- Best improvements are achieved when both dynamic phrase table and dynamic language model are employed. This is not surprising because they support each other in rewarding best or possibly new translation options suggested by the external user feedback. Moreover, each dynamic model is less effective if used alone, but still it is.
- The dynamic phrase table outperforms the hard-coded solution provided by Moses to insert new translation options via "-xml-input" feature. Through our novel approach Moses explores a larger search space and hence has an higher

level of flexibility to build the final translation; in fact, the cached translation options can be exploited at decoding time without any constraint. In the "xml-input" method they are instead strictly joint to a given source segment in a preliminary phase.
- The dynamic models are very effective with repetitive texts, but do not hurt with less repetitive texts; to a certain extent, accuracy gains are correlated with the document's level of repetitiveness.
- The dynamic models are absolutely agnostic about the informing data, which are provided from an external module (Suggestion Manager) and added into their caches without any quality check. Hence, the dynamic system strongly depends on the capability of this module to extract good, reliable and useful information from the user post-edits. In our ongoing further investigation, we observed much higher BLEU improvement, up to 20% relative, when a high-quality Suggestion Manager is employed.

The dynamic models were designed to store a limited amount of high-quality informing data (either phrase pairs or n-grams) and to make them available to the decoder in order to translate subsequent input sentences as coherently and consistently with the user feedback and the recent context as possible. By using time-decaying reward and penalty functions it is unnecessary to keep very old information in the cache, because their contribution in the overall translation process is negligible. In our experimentation, we usually stored no more than a thousand of entries. Indeed, in the considered online adaptation scenario, the recent user feedback is quite limited, hence the useful informing data which can be extracted from the feedback itself is small. If, by any chance, large amounts of informing data were available in advance, they could simply be exploited by other efficient but static data structures. Consequently, an extremely efficient data structure is not required at all; experimental evidence shows that the additional memory consumption is totally negligible in our standard configuration. By choice we preferred to have an easily adaptable and extendable implementation for the dynamic models.

We evaluated the impact of the dynamic models on the efficiency of the MT system by comparing the translation speed of the static and dynamic systems under computationally controlled conditions. We translated from English to Italian 10 documents from an Information Technology domain, for a total amount of 2.5K sentences and 39K running words, with the `"onlineSMT.py"` software introduced in Section 4. We employed the state-of-the-art static system[5] and the derived dynamic system equipped with a Suggestion Manager, which provides informing data by means of the Constrained Search algorithm.[6] With this setting, on the average approximately 7 phrase pairs and 53 n-grams are extracted from each post-edit and added to the dynamic

[5]For a fair comparison the static system also translates the text sequentially.

[6]Both systems are detailed in (Bertoldi et al., 2013).

phrase table and language model, respectively. The dynamic system is about 14% slower than the static system; approximately half of the additional computational time is due to the creation of the informing data by the Suggestion Manager. The dynamic system outperforms the static system by 13% relative BLEU.

7. Conclusion and Future Work

We have presented an enhanced version of Moses which provides a simple, modular and affordable solution to make it dynamic, in the sense that its models can be modified on-the-fly without reloading. The dynamic Moses system is hence able to quickly adapt to external feedback recommending new or better translation alternatives. This capability is achieved by means of a novel phrase table implementation and a new LM-like feature which dynamically store the external informing data and reward them according to a time-decaying scoring function.

These enhancements can be applied to the CAT scenario where the dynamic SMT system benefits from the translator's post-edits and s(he) benefits from the improved MT suggestions; hence, a virtuous cycle is established to diminish the overall post-editing effort. Recent papers presented experimental evidence of the effectiveness of the dynamic models described in this work; in an appropriate online MT adaptation framework the exploitation of the dynamic system significantly outperforms a static system up to 20% relative BLEU. In our experiments we found compatible results: the dynamic system outperforms the corresponding static system by 13% relative BLEU on the average, while its average translation speed decreases of approximately 14%.

However, these outcomes must be confirmed by employing the dynamic system in other translation tasks to further verify its versatility and its effectiveness. Additional experimental investigation is required to deeply understand the interaction between the components of the dynamic system, the effects of external factors, like the quality and the quantity of the informing data, the characteristics of the input text, and concurrent or contradictory feedbacks, and the impact of these variables on the overall performance.

The dynamic enhanced Moses version proposed in this work is under continuous development in order to further improve functions, efficiency and effectiveness and augment the field of applicability. Possible extensions are: (i) extending the dynamic language model to handle with cross-phrase n-grams; (ii) defining new scoring functions not related with recency, but to other variables, like human post-editors' confidence; (iii) enabling the contemporary usage of several dynamic phrase tables and language models; (iv) implementing a dynamic lexicalized reordering model mimicking the dynamic phrase table data structure. Some of these ideas are already under development, and they will be made available into Moses toolkit.

Acknowledgements

This research has been supported as part of the MosesCore project[7] (European Commission Grant Number 288487 under the 7th Framework Programme).

Special thanks got to Robert Grabowski, Liane Guillou, Michal Novák, Sorin Slavescu, José Camargo de Souza, with whom I worked during the MT Marathon 2012 to create the first implementation of the dynamic language model, and to Amin M. Farajian, who contributed to the development of the software supporting the dynamic system.

Bibliography

Bellegarda, Jerome R. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93 – 108, 2004.

Bertoldi, Nicola, Mauro Cettolo, and Marcello Federico. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proceedings of the MT Summit XIV*, pages 35–42, Nice, France, September 2013.

Cettolo, Mauro, Marcello Federico, and Nicola Bertoldi. Mining parallel fragments from comparable texts. In *Proceedings of the International Workshop on Spoken Language Translation*, Paris, France, 2010.

Federico, Marcello, Nicola Bertoldi, and Mauro Cettolo. IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models. In *Proceedings of Interspeech*, pages 1618–1621, Brisbane, Australia, 2008.

Gong, Zhengxian, Min Zhang, and Guodong Zhou. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919, Uppsala, Sweden, 2011. Association for Computational Linguistics.

Heafield, Kenneth, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Levenberg, Abby and Miles Osborne. Stream-based randomised language models for smt. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, pages 756–764, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[7] `http://www.mosescore.eu`

Levenberg, Abby, Chris Callison-Burch, and Miles Osborne. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, California, June 2010. Association for Computational Linguistics.

Nepveu, Laurent, Guy Lapalme, Philippe Langlais, and George Foster. Adaptive language and translation models for interactive machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 190–197, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Och, Franz Josef. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, 2003. Association for Computational Linguistics.

Stolcke, Andreas. Srilm - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, Denver, Colorado, 2002.

Talbot, David and Miles Osborne. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 7, pages 512–519, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Tiedemann, Jörg. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Wäschle, Katharina, Patrick Simianer, Nicola Bertoldi, Stefan Riezler, and Marcello Federico. Generative and discriminative methods for online adaptation in smt. In *Proceedings of the MT Summit XIV*, pages 11–18, Nice, France, September 2013.

Appendix A: Communication to the Dynamic System

A xml-based annotation[8] detailed in Tables 2 and 3 has been constructed to communicate with Moses to insert entries into the dynamic language model and phrase table. Note that the Moses parameter "-xml-input" should be set to "inclusive" to enable the parsing of the tag.

a_1	`<dlt cblm="Le visage rustre"/>`
a_2	`<dlt cblm="Le visage rustre de la domination || de la domination || visage"/>`
a_3	`<dlt cblm="Le visage rustre de la domination"/> <dlt cblm="de la domination"/> <dlt cblm="visage"/>`
b_1	`<dlt cblm-file="filename"/>`
b_2	`<dlt cblm-file="filename-1 || filename-2"/>`
c_1	`<dlt cblm-clear-entry="de la domination"/>`
c_2	`<dlt cblm-clear-entry ="de la domination || Le visage rustre || visage"/>`

c_3	`<dlt cblm-clear-entry ="de la domination"/>`
	`<dlt cblm-clear-entry ="Le visage rustre"/>`
	`<dlt cblm-clear-entry ="visage"/>`

d_1	`<dlt cblm-clear-all=""/>`
d_2	`<dlt cblm-command="clear"/>`

Table 2. Annotation to communicate with the dynamic language model. (a_1) insertion of one n-gram; (a_2) contemporary insertion of multiple n-grams; (a_3) sequential insertion of multiple n-grams; (b_1)-(b_2) insertion of n-grams from file(s); (c_1)-(c_3) deletion of single or multiple n-gram; (d_1)-(d_2) full cleanup of the dynamic language model. Double vertical lines separate multiple n-grams and filenames.

The tags can be included anywhere in the text to translate. Nevertheless, Moses first applies the required update to the dynamic language model and phrase table, then translates the actual input text with the updated models. If multiple tags are provided, they are exploited sequentially from left to right. If no text to translate is provided, Moses returns the translation of an empty string, in order to remain synchronized with the input.

The insertion of single or multiple entries are controlled by annotation in examples (a). If multiple entries are inserted with the annotation in examples (a_2), they are inserted contemporarily in the dynamic models and hence they will be associated to

[8] `dlt` stands for Document Level Translation because originally the dynamic models were intended for that task; `cblm` and `cbtm` stand for cache-based language model and translation model, respectively.

a_1	<dlt cbtm="The crude face ||| Le visage rustre"/>
a_2	<dlt cbtm="The crude face of domination ||| Le visage rustre de la domination |||| of domination ||| de la domination |||| face ||| visage"/>
a_3	<dlt cbtm="The crude face of domination ||| Le visage rustre de la domination"/> <dlt cbtm="of domination ||| de la domination"/> <dlt cbtm="face|||visage"/>
b_1	<dlt cbtm-file="filename"/>
b_2	<dlt cbtm-file="filename-1 || filename-2"/>
c_1	<dlt cbtm-clear-option="of domination ||| de la domination"/>
c_2	<dlt cbtm-clear-option="of domination ||| de la domination |||| The crude face ||| Le visage rustre |||| face ||| visage"/>
c_3	<dlt cbtm-clear-option="of domination ||| de la domination"/> <dlt cbtm-clear-option="The crude face ||| Le visage rustre"/> <dlt cbtm-clear-option="face |||| visage"/>
c_4	<dlt cbtm-clear-source="The crude face"/>
c_5	<dlt cbtm-clear-source="The crude face |||| of domination"/>
c_6	<dlt cbtm-clear-source"The crude face"/> <dlt cbtm-clear-source="of domination"/>
d_1	<dlt cbtm-clear-all=""/>
d_2	<dlt cbtm-command="clear"/>

Table 3. Annotation to communicate with the dynamic phrase table. (a_1) insertion of one translation option; (a_2) contemporary insertion of multiple translation options; (a_3) sequential insertion of multiple translation options; (b_1)-(b_2) insertion of translation options from file(s); (c_1)-(c_3) deletion of single or multiple translation options (c_4)-(c_6) deletion of all translation options associated to one or multiple source phrases; (d_1)-(d_2) full cleanup of the dynamic phrase table. Quadruple vertical lines separate phrase pairs; triple vertical lines separate source and target sides of a phrase pair. Double vertical lines separate multiple filenames.

a common age. Instead if separate tags for each entry are used, like in examples (a_3), the entries are inserted sequentially from left to right, and hence they receive different ages, with the right-most entry being the most recent.

The dynamic models can be also updated by loading entries and corresponding ages from one or more files, either in the initialization phase or with the annotation shown in examples (b). File formats for the dynamic language model and phrase table are shown in Table 4.

The deletion of single or multiple entries are controlled by annotations (c); while the overall cleanup of the models is done using the equivalent annotations in examples (d).

```
Le visage rustre ||| 1
la domination ||| 3
de la domination ||| 2
...|||...|||...
```

```
The crude face ||| Le visage rustre ||| 1
domination ||| la domination ||| 3
of domination ||| de la domination ||| 2
of domination ||| la domination ||| 3
...|||...|||...
```

Table 4. Format of the file containing informing data for updating the dynamic language model (above) and phrase table (below). The age is specified for each translation option and for each n-gram. In case of duplicates the last entry is valid.

Appendix B: Configuration of the Dynamic System

Assuming that the dynamic models are used as additional features in Moses, they must be specified in the configuration file, as explained below.

Moreover, the Moses parameter "-xml-input" should be set to "inclusive" to enable the parsing of the tags for updating the dynamic models. The Moses parameter "-no-cache" should be set to disable the caching of translation options; in fact, they and their scores can change over time as the dynamic models change.

Configuration of the Dynamic Language Model

The *feature* and *weights* sections must be modified to properly configure the dynamic language model and to set the corresponding weight.

```
[feature]
DynamicCacheBasedLanguageModel [parameters]

[weights]
DynamicCacheBasedLanguageModel0= <value>
```

The dynamic language model is configured in the *feature* section. The following parameters are currently configurable:

- `cblm-score-type`: scoring function type (see below); default is 0
- `cblm-query-type`: query type; default is 0
- `cblm-max-age`: maximum allowed age of the n-grams; default is 1000

The `cblm-score-type` parameter selects one of the age-dependent functions to score the n-gram stored in the cache of the dynamic language model. Table 5 shows the available scoring functions, which are described in Section 3.3. The `cblm-query-type` parameter selects one of the two modalities of lookup an n-gram in the cache of the dynamic language model described in Section 3.3 either 0 for *AllSubStrings* or 1 for *WholeString*. The `cblm-max-age` parameter sets the maximum age beyond which an n-gram is removed from the cache.

	scoring function type		
0	hyperbola-based reward	10	hyperbola-based penalty
1	power-based reward	11	power-based penalty
2	exponential-based reward	12	exponential-based penalty
3	cosine-based reward		

Table 5. Reward and penalty scoring functions implemented for the dynamic language model and phrase table.

The weight for the dynamic language model is set in the *weights* section.

Configuration of the Dynamic Phrase Table

The *feature*, *weights*, and *mapping* sections must be modified to properly configure the dynamic phrase table and the decoder and to set the corresponding weights.

```
[feature]
PhraseDictionaryDynamicCacheBased [parameters]

[weights]
PhraseDictionaryDynamicCacheBased0= <value>

[mapping]
0 T 0
1 T 1
```

The dynamic phrase table is configured in the *feature* section. As the dynamic phrase table extends the Moses basic object *PhraseDictionary*, all its parameters are still available. Moreover, few parameters are specific to the dynamic phrase table:

- `cbtm-score-type`: select the scoring function (see belov); default is 0
- `cbtm-max-age`: maximum allowed age of the phrase pairs; default is 1000

The `cbtm-score-type` parameter selects one of the scoring functions reported in Table 5 to score the phrase pairs stored in the cache of the dynamic phrase table. The `cbtm-max-age` parameter sets the maximum age beyond which a phrase pair is removed from the cache.

The weight(s) for the score(s) provided by the dynamic phrase table are set in the *weights* section. The number of values set in the *weights* section must coincide with the specified `num-features`.

Finally, in the *mapping* section the decoder must be informed about how many phrase tables are actually exploited. Assuming that the dynamic phrase table is used in addition to one standard phrase table, and that the translation options are fetched from either two, the section is set as shown above.

Address for correspondence:
Nicola Bertoldi
`bertoldi@fbk.eu`
Fondazione Bruno Kessler
via Sommarive 18, Povo, 38123 Trento, Italy

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 29–41

Integrating a Discriminative Classifier into Phrase-based and Hierarchical Decoding

Aleš Tamchyna[a], Fabienne Braune[b], Alexander Fraser[b], Marine Carpuat[c], Hal Daumé III[d], Chris Quirk[e]

[a] Charles University in Prague
[b] Ludwig Maximilians University Munich
[c] National Research Council Canada
[d] University of Maryland
[e] Microsoft Research

Abstract

Current state-of-the-art statistical machine translation (SMT) relies on simple feature functions which make independence assumptions at the level of phrases or hierarchical rules. However, it is well-known that discriminative models can benefit from rich features extracted from the source sentence context outside of the applied phrase or hierarchical rule, which is available at decoding time. We present a framework for the open-source decoder Moses that allows discriminative models over source context to easily be trained on a large number of examples and then be included as feature functions in decoding.

1. Introduction

Phrase-based and hierarchical SMT represent the state of the art for many language pairs. Both of these methods model translation through decomposing the input into segments (phrases, or source sides of hierarchical rules) and translating each one separately. In other words, the translation units are considered independent of each other. Generally, these are scored using relative frequencies. This strong independence assumption often prevents phrase-based or hierarchical models to correctly choose between ambiguous segments. As an example, consider the polysemous French noun-phrase "un rapport" where "un" is the masculine indefinite article in French, and "rapport" is a noun which can mean "report" or "relationship". Figure 1 shows sam-

© 2014 PBML. All rights reserved. Corresponding author: tamchyna@ufal.mff.cuni.cz
Cite as: Aleš Tamchyna, Fabienne Braune, Alexander Fraser, Marine Carpuat, Hal Daumé III, Chris Quirk. Integrating a Discriminative Classifier into Phrase-based and Hierarchical Decoding. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 29–41. doi: 10.2478/pralin-2014-0002.

ple training data with 2 parallel sentences (let us assume that the word alignment is correct, i.e., 1-to-1 and monotonic in both cases).

Sentence 1: il a rédigé un **rapport** .	he has written a **report** .
Sentence 2: un **rapport** entre les coûts et ...	a **relationship** between the costs and ...

Figure 1. Example parallel data for an SMT system.

For this data, the maximum likelihood estimates for "un rapport" - "a relationship" and "un rapport" - "a report" are both 0.5, since they both occur once. If we then observe a test sentence containing "rédigé" (to write), the direct phrasal translation feature function is unable to distinguish between the two translations of "un rapport", although it is clear that the "report" sense should be more probable. The only component in the translation model able to perform some disambiguation in this case is the language model, which is typically employed to ensure the coherence of MT output. However, it only has a limited scope and its estimation suffers from data sparsity when the window size is increased. Consequently, phrase-based and hierarchical models leave space for improving translation quality by conditioning the choice of translation units on contextual information. There have indeed been a number of successful attempts to exploit context on the source (input) side.

In this work, we integrate a discriminative classifier into the open-source decoder Moses in order to score translation rules using richer models of their source context. Related work on discriminatively trained word and phrase lexica will be presented and discussed in Section 5. We provide a complete framework for both phrase-based and hierarchical translation that allows the training of discriminative models over source-side context and the inclusion of classifier predictions in decoding.

The paper is organized as follows: Section 2 describes the relationship between SMT and discriminative classification and provides details of our machine learning setting. In Section 3, we describe the integration into Moses, including the interface for defining new classifier features. Section 4 discusses our experiments. Section 5 concludes the paper with a discussion.

2. Discriminative Classification

We have integrated the high-speed streaming classifier Vowpal Wabbit (VW) into Moses to act as a discriminative phrase lexicon. In this section, we first present the integration of our discriminative model into the Moses translation model. In a second step, we discuss how VW works as discriminative classifier, using a set of label-dependent features. Finally, we show how to train our model using VW.

2.1. Integration into the Moses Translation Model

In Moses, the translation model is implemented as a so-called log-linear model, see Formula 1, which is a linear model that uses feature functions which often look like log probabilities.

$$p_\lambda(E, A, S|F) \propto \exp(\sum_i \lambda_i h_i(A, S, E, F)) \quad (1)$$

In Formula 1, h_i are feature functions and λ_i are the corresponding weights. The formula expresses the probability of sentence E, full sentence word alignment A, and source phrasal segmentation S, given source sentence F. We use lowercase e, f to denote phrases, and a to denote a phrasal word alignment. Feature functions in current (phrasal or hierarchical) SMT systems are typically dense, i.e., they output a small, fixed number of features (feature scores). Baseline features include direct and inverse phrase translation probabilities $p(e, a|f), p(f, a|e)$, lexical weights $p_{lex}(e, a|f)$, $p_{lex}(f, a|e)$, the English language model score $p(E)$ (often implemented as a 5-gram) and others. The conceptually most important feature function in Moses is the feature function modeling $p(e, a|f)$, where f is a source-language phrase, e is a target-language phrase and a is the word alignment between e and f. This is often referred to as the direct phrasal translation probability.

We augment the translation model given in Formula 1 with a new feature function conditioning not only on the source-language phrase but also on the context around it. Formally, we define $p(e, a|f, f')$, where f' represents the input source-language sentence context external to the source-phrase f being translated by this phrase pair. We use discriminative classification to compute this probability because it allows us to use arbitrary information on the source side. We also take advantage of factored machine translation by overloading f and e using factors (we use one factor of morphological tags, and one factor of lemmas). Finally, we allow the external source-side context f' to also access the same factored information (tags and lemmas) in the source sentence external to the phrase being translated. We will sometimes abuse notation to drop a.

2.2. VW as a Discriminative Phrase Lexicon Model

We train VW to choose a target-language (English) phrase e and a word alignment a given a source-language (French) phrase f and all other information available about the source-language sentence F, which we denote f'. We accomplish this by training the classifier on examples extracted from each aligned phrase-pair instance (e, f) in the automatically word-aligned parallel training data. For a new input sentence and particular source-language phrase f, we would like to choose the correct target-language phrase e based on the similarity of f and f' to examples in the training data. To accomplish this, we use label-dependent classification, as outlined below.

Label-dependent Features. In the Discriminative Phrase Lexicon scenario we wish to estimate the probability for a target-language phrase e given a source-language phrase f and the source sentence external context f'. We therefore associate one set of features used for classification with the fixed source phrase and source external context, and have another set of features which varies with the target-language phrase e (i.e., label-dependent). We define the feature space as the $S \times T$ cross-product – this is similar to simply concatenating each source feature with each target feature. We also take the features themselves without concatenation. Figure 2 shows the implemented features on a sample sentence. By taking the cross-product of the source and target features, we obtain a powerful final representation for classification. For instance, we implement the featurization of a discriminative word (rather than phrase) lexicon by taking the cross-product of source-context-bag-of-words and source-phrase-bag-of-words with target-phrase-bag-of-words.

Context

Form:	nous	ne	le	savons	pas	encore	.
Lemma:	il	ne	le	savon	pas	encore	.
Tag:	CLS	ADV	DET	NC	ADV	ADV	PONCT

Phrase Pair

Source:	ne le savons pas
Target:	do not know
Alignment:	0-1 1-2 2-2 3-1
Scores:	-7.5 -9.2 -1.6 -7.5

Features

Source indicator:	pˆne_le_savons wˆpas
Target indicator:	pˆdo_not_know
Source internal:	wˆne wˆle wˆsavons wˆpas
Target internal:	wˆdo wˆnot wˆknow
Context:	cˆ0_-1_nous cˆ1_-1_il cˆ2_-1_CLS cˆ0_1_encore ...
Paired:	pˆne_not pˆle_know pˆsavons_know pˆpas_not
Scores:	scˆ0_-10 scˆ0_-9 scˆ0_-8 scˆ1_-10 scˆ2_-10 ...

Figure 2. Implemented Features

Vowpal Wabbit. To implement label-dependent classification, we chose to use Vowpal Wabbit (VW),[1] implemented by John Langford. VW has a fast implementation of stochastic gradient descent and L-BFGS for many different loss functions. VW is

[1] `http://hunch.net/~vw/`

widely used for machine learning tasks. VW was built into a library for the work reported here. VW has built-in support for:

- Feature hashing (scaling to billions of features)
- Caching (no need to re-parse text)
- Different losses and regularizers
- Reductions framework to binary classification/regression
- Multithreading/multicore processing.

2.3. Training the discriminative Model

For each (e, f) instance in the file with extracted phrase pairs, we create a training example. We first perform significance testing (Johnson et al., 2007) to reduce the total number of (e, f) types in the phrase table. We then extract one training example per (e, f) instance extracted. We generate a line for each possible translation of f in the reduced phrase table. The correct translation is assigned a *loss* of zero, all other translations of f get a loss of 1.

Example. As an example, consider the polysemous French noun-phrase "un rapport" introduced in Section 1, which can either be translated as "report" or "relationship". We have shown that although the translation model cannot adequately choose between these translations, it is clear that in a test sentence containing "rédigé" (to write), the "report" translation should be more probable. We can operationalize this with label-dependent features as shown in Figure 3.

	Source Namespace	**Target Namespace**	**Loss**
Sentence 1:	p^un_rapport c^il c^a **c^rédigé**	p^a_report	0
		p^a_relationship	1
Sentence 2:	p^un_rapport c^entre c^les ...	p^a_report	1
		p^a_relationship	0

Figure 3. Training examples with label-dependent features extracted from sample parallel data.

The features prefixed with "p^" are the phrases being modeled. The features prefixed with "c^" implement the bag-of-words feature. Implementation details of the training procedure are given in the next section. In our example, during training the model can learn from Sentence 1 that the S × T cross-product feature c^rédigé.p^a_report should push the loss towards zero. At testing, this allows "a report" to be chosen.

We train VW using the cost-sensitive one-against-all reduction and label-dependent features. We use the dev set (the same dev set as is used for MERT) to perform early stopping.

3. Integration of VW into Moses

In this section, we present the engineering details of the integration of VW into the phrase-based and hierarchical components of Moses. As an overall illustration, we compare the standard Moses pipeline and a pipeline with integrated classifier. To this aim, consider again the polysemous French noun-phrase "un rapport" presented in Section 1. In the standard pipeline, shown in Figure 4 (left), phrases or hierarchical rules are extracted from the word-aligned parallel data and scored using maximum likelihood estimation.

During decoding, the scored units are applied. As noted in sections 1 and 2, this pipeline does not allow to choose the correct translation of "un rapport" given the source sentence context. In a pipeline where VW is integrated to Moses, shown in figure 4 (right), the training procedure is augmented with an additional step to train the discriminative model using VW. The details of classifier training have been presented in Section 2.3. During decoding, the trained model is queried and the obtained prediction is added to the log-linear model, as shown in Section 2.1. This additional score allows the system to choose the correct translation of "un rapport" given the source context.

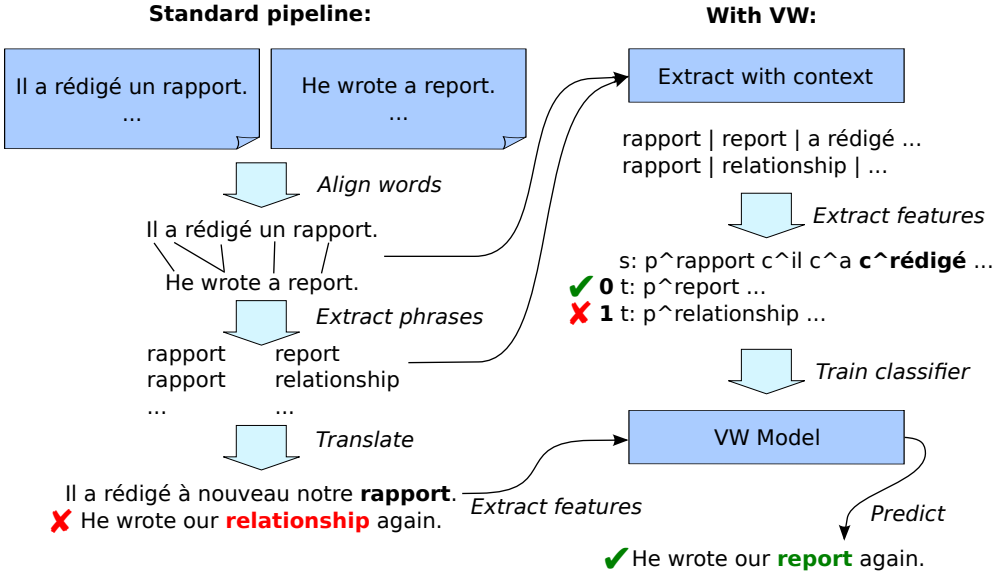

Figure 4. Classifier Pipeline. Training examples are extracted along with context which helps disambiguate phrasal translations.

We integrate VW in training and decoding. For both tasks an interface between the MT system (Moses) and the learner (VW) has to be created.

We begin by presenting the overall architecture of the interface between Moses and VW in Section 3.1. Then we explain how to train such a model in Section 3.2 and query it during decoding in Section 3.3.

3.1. Overall Architecture

We integrate the learner (VW) into the MT system (Moses) by defining a single library used for training the discriminative model as well as getting model predictions during decoding. Using this library avoids code duplication but even more importantly assures consistent definition and configuration of features for model training and prediction. Also, it makes the overall architecture simple and extensible. The design of the library is inspired by the Producer/Consumer pattern: we decouple (i) feature extraction from training and input data and (ii) generation of features provided to the learner. (i) is implemented by the *Feature Extractor* interface and (ii) by the *Feature Consumer* interface. The design chosen also allows us to clearly separate the logic of feature generation from the specificities of the learner. The only requirement needed to add different learners is the implementation of the *FeatureConsumer* interface. For instance, it would be easy to add such an interface for MegaM,[2] which we leave for future work.

The *FeatureExtractor* interface supports extraction of various types of features from the source sentence context as well as from the source and target side of phrases (for phrase-based SMT) or hierarchical rules (for hierarchical SMT). Feature extraction is controlled through the specification of a configuration file. The current implementation supports the features presented in Section 2.2.

We provide 3 implementations of the *FeatureConsumer*. The two first are used to train the discriminative model and the third to get model predictions. More precisely, the *VWFileTrainConsumer* extracts features for training VW in text format and stores them in an output file. We use this class in training to visually inspect the generated features before training the VW model. Otherwise, it is possible to feed the features directly into VW and train the model using *VWLibraryTrainConsumer*. In decoding, *VWLibraryPredictConsumer* is used to get VW predictions.

The same library is implemented in the phrase-based and in the hierarchical components of Moses.

3.2. Training the Model

For training our discriminative model, we first modify the phrase and hierarchical rule extraction algorithms provided in the phrase-based and hierarchical components

[2] `http://www.umiacs.umd.edu/~hal/megam/`

of Moses. The modified routines output an additional file indicating in which source sentence and at which position each phrase and rule have been extracted. These annotations allow us to extract "source sentence context" features for each phrase and hierarchical rule. In order to provide a richer context, the source side training data is augmented with a factored annotation containing morphological and POS tags. Using these annotations as well as the parallel corpus, training examples for VW are extracted using the feature library described in Section 3.1. The *FeatureExtractor*, controlled by the configuration file, specifies which features are extracted while the *FeatureConsumer* generates the training examples for VW.

3.3. Getting Predictions during Decoding

Model predictions queried during decoding are implemented as a feature function which is added to the log-linear model and consequently tuned on a held-out data set. In its phrase-based and hierarchical components, Moses offers an interface for defining feature functions. Our model predictions are integrated into the decoder by implementing this interface (with some tricks). The feature function for the phrase-based component is located in the class *ClassifierFeature* while the feature function for the hierarchical component is in *ContextFeature*.

In the phrase-based as well as the hierarchical component of Moses, the task of the feature function is to re-evaluate each translation option by querying VW according to the source sentence context in which they occur. More precisely, for each translation option applying to a given span, source context features are extracted using the library described in Section 3.1.

Using the extracted features, the *VWLibraryPredictConsumer* is used to query the VW model. The obtained predictions are then normalized to transform model scores (losses) into probabilities. The feature function is evaluated prior to phrase- or rule-table pruning and decoding. This allows us to save computation and to avoid discarding options which our feature considers good.

Note that in order to implement this behavior, we needed to deviate a little from the standard feature interface: our feature function is called immediately after translation options are collected and evaluates all translation options for a given source span at the same time (to allow normalization).

3.4. Integration into the Hierarchical Component

Integration of a discriminative model into a hierarchical system is generally more challenging than the integration into a phrase-based system. The main reasons are that (i) many left-hand-sides of hierarchical rules can apply to the same source sentence span and (ii) a single left-hand-side of a hierarchical rule can apply to many source sentence spans. As a consequence, many more rules have to be collected to generate training examples and many more translation options have to be re-evaluated during decoding.

As an illustration, consider the French segment "patiente diabétique et enceinte" (an English word-by-word gloss is "patient diabetic and pregnant", it means "diabetic and pregnant patient"). Consider that the following hierarchical rules have been extracted from the training data.

r1 $X/X \rightarrow$ <X_0 enceinte , pregnant X_0>
r2 $X/X \rightarrow$ <X_0 enceinte , X_0 enclosure>
r3 $X/X \rightarrow$ <patiente X_0 , X_0 patient>
r4 $X/X \rightarrow$ <patiente X_0 et X_1, X_0 and X_1 patient>

Each rule consists of lexical items and aligned non-terminal symbols. All rules presented above match the source segment "patiente diabétique et enceinte" although rules r1, r3 and r4 have different source language sides. On the other hand, rules with source side "X enceinte" can apply to the complete segment "patiente diabétique et enceinte" or to the last three or two words.

As shown in Section 2.2, features extracted to train and query the integrated model include source context features. The extraction of these features requires information about the source sentence context surrounding the place where a rule has been extracted (training) or applied (decoding). In a phrase-based system, the source context is the context surrounding a single source phrase. In a hierarchical system, the source context cannot be attached to a single source side of rule because several rules can match the same source language segment. Hence computing a discriminative score for hierarchical rules according to their context of occurrence in the source language sentence requires to collect, for a given span, all hierarchical rules applying to this span. For instance when considering the context surrounding the linguistic segment "patiente diabétique et enceinte" all rules applying to the span beginning at "patiente" and ending at "diabetique" have to be collected. Such rules include rules r1,r2 and r3.

4. Experiments

Performance. In terms of performance, the phrase-based component of Moses with the discriminative model takes 80% relative longer than the Moses baseline without VW. The hierarchical component is slower (300% relative longer) due to the additional complexity described in Section 3.4.

We made queries to VW thread-safe and tested all of our code in a parallel setting.

The classifier feature is also fully integrated in Moses' Experiment Management System (EMS, experiment.perl) which allows potential new users to quickly create experiments with our feature.

Phrase-based Experiments. We used our feature in a setting similar to phrase-sense disambiguation (PSD, Carpuat and Wu, 2007), utilizing all of the classifier features described in Section 3.1. We utilized science domain training data consisting of

113291 French and English parallel sentences, as well as dev and test sets distributed with this data.[3] We computed GIZA++ word alignments by using a much larger parallel French/English corpus of over 2 million parallel sentences (this was only used to improve the word alignment). We trained a 5-gram language model using SRILM and decoded using KenLM. Table 1 shows the results of our experiment. Our feature receives a moderately high weight (the weight of the direct phrasal translation probability is 0.10). Our integrated system beats the baseline by a difference of 0.60 BLEU.

Source	Target	BLEU		Feature
		Baseline	+Classifier	Weight
French	English	32.62	33.22	0.05

Table 1. Results of experiment with phrase-based translation

Hierarchical Experiments. The same experiment has been conducted using the hierarchical component of the system. Table 2 shows the results. Our feature receives a high weight and the integrated system beats the baseline by a difference of 0.53 BLEU. Overall, the hierarchical system performs worse than phrase-based for this experiment.

Source	Target	BLEU		Feature
		Baseline	+Classifier	Weight
French	English	31.08	31.61	0.14

Table 2. Results of experiment with hierarchical translation

5. Discussion and Conclusion

In most SMT architectures, translation rules are scored based on their relative frequency in the parallel training corpus. However, integrating richer information into translation decisions is an active area of research.

We integrated a discriminative classifier into Moses in order to score translation rules using richer models of their source context. This contrasts with the feature-rich approaches already available in Moses. For instance, factored translation models (Koehn and Hoang, 2007) can be used to define translation rules based on lemma, POS, or other representations of phrases, but these rules are still scored using relative

[3] `http://www.umiacs.umd.edu/~hal/damt/`

frequency. Source context features, such as words and part-of-speech tags surrounding a given source phrase, can also be directly made available to the decoder as features in the log-linear model (Gimpel and Smith, 2008). While this approach presents the advantage of directly optimizing feature weights for BLEU or other metrics of translation quality, it suffers from current limitations with large-scale discriminative training of SMT systems, as discussed in Section 2.

Our approach is inspired by context-dependent *phrase* lexicons for phrase-based SMT models (Giménez and Màrquez, 2007; Stroppa et al., 2007; Carpuat and Wu, 2007; Weller, 2010; Haque et al., 2011). These models generalize early discriminative word translation models (Berger et al., 1996) to current phrase-based SMT. Unlike in Berger et al. (1996), our phrasal translations are conditioned on the observed source sentence context, rather than on the hypothesized target language context, which facilitates the integration of context on the source side, such as the local and long-distance clues used in word sense disambiguation.

Other work has focused on *word*-level discriminative lexicons (Bangalore et al., 2007; Mauser and Ney, 2009; Venkatapathy and Bangalore, 2009), which predict which words should occur in the output sentence based on a bag-of-words representation of the source sentence. Extensions include Niehues and Waibel (2013) who enrich the bag-of-word context with n-grams and syntactic features, and Jeong et al. (2010), who focus on translation into morphologically richer languages.

Context-dependent scoring of translation rules can make decoding significantly more expensive, as context-dependent translation probabilities cannot be pre-computed once and for all at training time. Previous experimental implementations expanded phrase-tables to represent phrase instances rather than types (Giménez and Màrquez, 2007). In contrast, we integrate a fast online classifier designed for large data directly into the decoder. Our implementation uses a single global model for disambiguating all source phrases, rather than one model per phrase type as in Carpuat and Wu (2007); Giménez and Màrquez (2007) and in the Moses implementation of Mauser and Ney (2009).[4]n addition to making the implementation and training easier and more efficient, this approach can also potentially capture generalizations across phrase types as observed for word lexicons in Jeong et al. (2010).

In hierarchical phrase-based SMT, classifiers have been used to disambiguate between translations of words and short phrases (Chan et al., 2007), and to model reordering decisions (Xiong et al., 2006). In contrast, our implementation lets us directly score each translation rule, and can simultaneously be used to model soft syntactic constraints and lexical disambiguation clues.

Our work is also somewhat related to so-called sparse features (implemented in Moses by Hasler et al., 2011). The weights of these features are trained on the development set using MIRA. This is advantageous, as the weights of features associated with the development set are optimized to maximize the final performance criterion

[4] http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc32

(BLEU) directly. However, the development set is typically very small, which limits the coverage of the features which can be effectively trained. We avoid the problem of small dev sets by training our VW model on the training set. Our feature is then just one score in the log-linear model.

We have integrated VW into Moses and provided a proof-of-concept implementation of a discriminative phrase lexicon. In the future we plan to implement other feature functions and integrate other classifiers. All of our code is publicly available in the Moses repository in the branches *damt_phrase* and *syntaxContext*.

Acknowledgement

This work was partially supported by Deutsche Forschungsgemeinschaft grant Models of Morphosyntax for Statistical Machine Translation (Phase 2), by NSF Grant No 1005411, by DARPA CSSG Grant D11AP00279, by grant FP7-ICT-2011-7-288487 (MosesCore) of the European Union, by project LH12093 of the Ministry of Education, Youth and Sports of the Czech Republic, and by SVV project number 260104.

A. Installation

1. Download and compile VW:

```
git clone https://github.com/JohnLangford/vowpal_wabbit.git
cd vowpal_wabbit
./autogen.sh --prefix=`pwd`
make && make install
```

2. Download and compile Moses:

```
git clone https://github.com/moses-smt/mosesdecoder.git
cd mosesdecoder
git checkout damt_phrase # or syntaxContext for hiero
./bjam --with-vw=<path-to-vowpal-wabbit>
```

3. See the provided sample EMS configuration file and INI file for the VW feature function:

```
mosesdecoder/scripts/ems/example/config.psd
mosesdecoder/scripts/ems/example/data/psd-features.ini
```

Bibliography

Bangalore, Srinivas, Patrick Haffner, and Stephan Kanthak. Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction. In *ACL*, 2007.

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

Carpuat, Marine and Dekai Wu. Improving statistical machine translation using word sense disambiguation. In *EMNLP*, 2007.

Chan, Yee Seng, Hwee Tou Ng, and David Chiang. Word Sense Disambiguation Improves Statistical Machine Translation. In *ACL*, 2007.

Giménez, Jesús and Lluís Màrquez. Context-aware Discriminative Phrase Selection for Statistical Machine Translation. In *WMT*, 2007.

Gimpel, Kevin and Noah A. Smith. Rich source-side context for statistical machine translation. In *WMT*, 2008.

Haque, Rejwanul, Sudip Kumar Naskar, Antal Bosch, and Andy Way. Integrating source-language context into phrase-based SMT. *Machine Translation*, 25(3), 2011.

Hasler, Eva, Barry Haddow, and Philipp Koehn. Margin infused relaxed algorithm for Moses. *Prague Bull. Math. Linguistics*, 96, 2011.

Jeong, Minwoo, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. A discriminative lexicon model for complex morphology. In *AMTA*, 2010.

Johnson, Howard, Joel Martin, George Foster, and Roland Kuhn. Improving Translation Quality by Discarding Most of the Phrasetable. In *Proc. of EMNLP-CoNLL 2007*, 2007.

Koehn, Philipp and Hieu Hoang. Factored translation models. In *EMNLP*, 2007.

Mauser, Arne and Hermann Ney. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *EMNLP*, 2009.

Niehues, Jan and Alex Waibel. An MT error-driven discriminative word lexicon using sentence structure features. In *WMT*, 2013.

Stroppa, Nicolas, Antal van den Bosch, and Andy Way. Exploiting source similarity for SMT using context-informed features. In *TMI 2007*, 2007.

Venkatapathy, Sriram and Srinivas Bangalore. Discriminative machine translation using global lexical selection. *TALIP*, 8(2), May 2009.

Weller, Marion. An empirical analysis of source context features for phrase-based statistical machine translation. Diploma thesis, Universität Stuttgart, 2010.

Xiong, Deyi, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In *ACL*, 2006.

Address for correspondence:
Aleš Tamchyna
tamchyna@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 101 APRIL 2014 43–54

Visualization, Search and Analysis of Hierarchical Translation Equivalence in Machine Translation Data

Gideon Maillette de Buy Wenniger, Khalil Sima'an

Institute for Logic Language and Computation (ILLC), Faculty of Science, University of Amsterdam

Abstract

Translation equivalence constitutes the basis of all Machine Translation systems including the recent hierarchical and syntax-based systems. For hierarchical MT research it is important to have a tool that supports the qualitative and quantitative analysis of hierarchical translation equivalence relations extracted from word alignments in data. In this paper we present such a toolkit and exemplify some of its uses. The main challenges taken up in designing this tool are the efficient and compact, yet complete, representation of hierarchical translation equivalence coupled with an intuitive visualization of these hierarchical relations. We exploit a new hierarchical representation, called Hierarchical Alignment Trees (HATs), which is based on an extension of the algorithms used for factorizing n-ary branching SCFG rules into their minimally-branching equivalents. Our toolkit further provides a search capability based on hierarchically relevant properties of word alignments and/or translation equivalence relations. Finally, the tool allows detailed statistical analysis of word alignments, thereby providing a breakdown of alignment statistics according to the complexity of translation equivalence units or reordering phenomena. We illustrate this with an empirical study of the coverage of inversion-transduction grammars for a number of corpora enriched with manual or automatic word alignments, followed by a breakdown of corpus statistics to reordering complexity.

1. Introduction

What kind of translation equivalence occurs in the word aligned data of a language pair? Are the word alignments yielding them correct and what kind of requirements for effective translation models is implied by their complexity? Most practitioners of

© 2014 PBML. All rights reserved. Corresponding author: gemdbw AT gmail.com
Cite as: Gideon Maillette de Buy Wenniger, Khalil Sima'an. Visualization, Search and Analysis of Hierarchical Translation Equivalence in Machine Translation Data. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 43–54. doi: 10.2478/pralin-2014-0003.

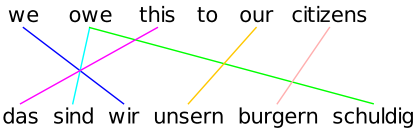

Figure 1: Example of alignment visualization for an aligned sentence pair (Europarl)

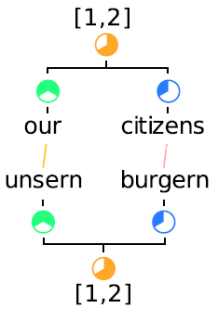

Figure 2: Example recursive composition of HATs

Machine Translation have some qualitative appreciation of the difference in similarity across language pairs. For example, it is a well known fact that for Chinese-English translation long range reordering is the norm, and as a result phrase-based translation does not work well for this language pair. Similarly it is known, that reordering is more local for Arabic-English than for Chinese-English, so that phrase-based translation gives relatively good performance for this language pair. Qualitative knowledge of the data is much more scarce for less popular language pairs and less famous but nevertheless valid reordering constructions; and in fact hard statistics about alignment complexity are mostly lacking for all language pairs. Crucial for the success of machine translation are the quality of the aligned corpus as operationalized by induced translation equivalence, and its complexity and compatibility with a translation model. A tool that facilitates more knowledge of these factors and that gives better qualitative understanding of translation equivalence occurring in real data can be valuable. A crucial aspect of such a visualization tool is an efficient and compact representation of hierarchical translation equivalence coupled with an intuitive visualization of the hierarchical relations between the translation equivalents. We employ Hierarchical Alignment Trees (HATs) (Sima'an and Maillette de Buy Wenniger, 2013) as the representation of choice for our *HATs toolkit*.

The word alignment in Figure 1 exemplifies the fact that manual or mental reconstruction of a representation of hierarchical translation equivalence over word alignments can be impractical for more than a few word long sentences with complex alignments, which constitutes the actual scenario when doing analysis of word alignments for representative, large data.

Besides visualization, one other application of the HATs toolkit is *search and data analysis*. Based on properties of the HATs representation it is easy to answer a question like "are their many valid/correct complex alignment constructions that cannot be covered by Inversion Transduction Grammars (ITG) in big automatically aligned corpora?". Previous work on alignment coverage (Zens and Ney, 2003; Wellington

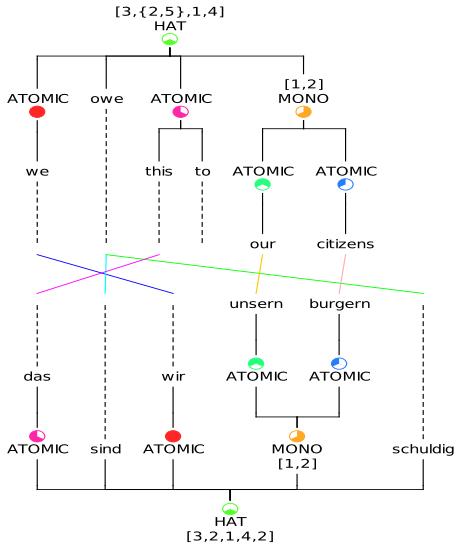

Figure 3: Visualization aspects: Filling (and color) of the circles indicates the equivalence between the top source-to-target HAT and the mirrored target-to-source HAT displayed below it. Labels, such as [3,{2,5},1,4] at the top node, denote permutation-set reordering operations at nodes. The labels *ATOMIC*, *MONO* and *HAT* in the HAT visualization indicate broad complexity categories for reordering.

et al., 2006; Søgaard and Wu, 2009; Søgaard and Kuhn, 2009; Søgaard, 2010) has been mainly involved with hand-aligned corpora, and as such has not given much direct insight into the validity of claims like "many of the non-binarizable alignments in automatically aligned data correspond to alignment errors". Our toolbox provides a method for automatically compiling a database of alignment properties for an input parallel corpus, so that alignments with certain properties can be effectively queried and retrieved. This makes it very simple to select for example all non-binarizable alignments. Based on such selections and by a quick visual inspection of the associated HATs, researching the validity of claims about the complexity of word alignments in real data becomes far simpler than building specific tools for ITG (or any other specific synchronous grammar formalism). Quantitative analysis in the form of statistics like alignment coverage for different types of permissible reordering complexity is another important component in the more global analysis of translation equivalence properties. For such quantitative analysis, keeping track of the reordering operations at the nodes while at the same time computing the HATs in an efficient and compact way will prove essential. We will exemplify this by reporting extensive empirical analysis of the coverage of ITG on a range of parallel corpora for manual as well as automatic word alignments. To the best of our knowledge, these results are the first on *large, automatically aligned* parallel corpora. In the remainder of this paper we first introduce HATs and discuss some of their visualization properties, then briefly discuss the methods available in our tool and move quickly to the empirical study regarding binarizable word alignments that can be covered by ITG.

2. Hierarchical Alignment Trees (HATs)

An efficient and compact representation of hierarchical phrase pair translation equivalence is **Hierarchical Alignment Trees (HATs).** A **HAT** (Sima'an and Maillette de Buy Wenniger, 2013) is a hierarchical representation/factorization of the phrase pairs in a word alignment; A HAT compactly represents a synchronous tree pair: a source and target tree pair with a node alignment relation between them. The recursive structure in a HAT shows the build up of phrase pairs from embedded phrase pairs. Hence, every node in a HAT represents the phrase pair at the fringe of the subtree under that node.

In Figure 3 we show what a visualization of a *Hierarchical Alignment Tree* for the example of Figure 1 looks like. There are actually *two* HATs displayed here. The upper tree shows the mapping from source to target while the bottom tree shows the mapping from target to source. Between the two trees we display the word alignments, making it directly clear how certain parts of the word alignment yield corresponding parts in the HATs. The filling and color/shade of the nodes clearly represents the translation equivalence between the source and target side of phrases.

Like **Normalized Decomposition Trees (NDTs)** (Zhang et al., 2008), HATs are *minimally branching factorization* of word alignments into phrase pairs, i.e., every HAT node covers a phrase pair and it dominates the smallest number of translation units (the child nodes) that the phrase pair decomposes into. In Figure 3, the alignment underlying the phrase pair *our citizens unsern burgern* decomposes down minimally to two phrase pair nodes *our unsern* and *citizens burgern*. More intricate HATs may arise due to complex word alignments that involve many-to-many and discontiguous translation units.

Discontiguous translation units One important property of HATs is their explicit representation of discontiguous translation units. In Figure 3 the root node on the English/German side dominates, among others, two terminal nodes: this explicitly represents the discontiguous unit given by the alignment between *sind + schuldig* (positions 2 and 5) on the German side with a single English word *owe* (position 2). More generally, to differentiate between phrase pairs and separate parts of discontiguous translation units in the HAT representation, the latter are depicted as *terminal nodes* (nodes labeled with words without any children), whereas the former (phrase pairs) are represented as non-terminal nodes (circles with filling dominating a subtree).

Reordering operators Crucially, HATs extend **NDTs** by providing explicit representation of the reordering of the children under every node by a transduction operator, called a *set-permutation*, as well as the internal word alignments for atomic (non-decomposable) phrase pairs. Hence, a HAT is a decorated tree where the nodes

are decorated with *set-permutations*. A set-permutation decorating a node in source side HAT is a list of integer sets denoting a transduction operation that applies to the children of that node to obtain the target side phrase reordering. We will immediately exemplify HATs and set-permutations before we proceed further with discussing the properties of HATs.

Set-permutations such as [3,{2,5},1,4] (see the root node in Figure 3) denote reordering operations occurring under these nodes. In this example the source children 1, 2, 3, 4 map to target children (relative order) 3, {2,5}, 1, 4 respectively, where {2, 5} represents the fact that the second child is linked with two on the other side in positions 2 and 5. In the simpler monotone mapping *our citizens | unsern burgern* the set-permutation label is [1,2]. We also see the coarse reordering categories *ATOMIC*, *MONO* and *HAT* in this figure, which are discussed next.

Complexity categories As mentioned above, every node is decorated with a set-permutation, specifying the relative mapping occurring directly below it. In the case of bijective mappings this describes a permutation. In the general case of arbitrary m-n mappings there are recurring target position in the mapping set of different source positions and/or multiple target positions occurring in the mapping set(s) of some source positions. Hence, the set-permutations can be grouped into coarser categories of mapping complexity. We distinguish the following five cases, ordered by increasing complexity:

1. *Atomic*: If the alignment does not allow the existence of smaller (child) phrase pairs: a subset of alignment positions that is not connected to the other positions while also forming a contiguous sequence on the source and target does not exists.
2. *Monotonic*: If the alignment can be split into two monotonically ordered parts.
3. *Inverted*: If the alignment can be split into two inverted parts.
4. *PET (Permutation Tree)*: If the alignment can be factored as a permutation of more than 2 parts.
5. *HAT (Hierarchical Alignment Tree)*: If the alignment cannot be factored as a permutation of parts, but the phrase does contain at least one smaller phrase pair.

Typically there are multiple HATs for a word alignment, corresponding to different possible minimally branching factorizations into phrase pairs. These alternative HATs can be efficiently computed and stored as a chart using a CYK-parser like chart parsing algorithm that parses the alignment and builds a Hypergraph of HATs in the process.[1]

[1]This is exactly what is done by our program. Note that in certain cases the number of HATs per alignment can become big, in particular for alignments that contain many monotone parts. One optimization we use in our algorithm is reasoning about null-aligned words outside the main algorithm. Computation is typically fast, provided enough memory is available. Rendering all HATs is done by

A categorization of the complexity of the HAT as a whole is determined based on the complexity categories of the alignment mappings at its nodes. *Binary Inversion-Transduction Trees (BITTs)* is the least complex class consisting of only binary HATs that can be built for binarizable permutations (Huang et al., 2009), any HAT that contains only Monotonic and/or Inverted nodes belongs to this class. If a HAT contains at least one PET node but no HAT nodes it belongs to the category called PETs corresponding to general permutations (Zhang et al., 2008; Satta and Peserico, 2005). Finally the occurrence of at least one HAT node implies the set *HATs* which captures all possible many-to-many mappings.

Having broadly explained what Hierarchical Alignment Trees (HATs) are about and what kind of information about hierarchical translation equivalence they give, the next question is what other things we can do with HATs apart from this most basic form of visualization.

3. Empirical analysis of word alignments in parallel corpora

In this section we exemplify one use of our toolkit for analysis of translation units (TUs) and word alignments in parallel corpora. We first look at the percentage of word alignments covered by Inversion-Transduction Grammar (ITG) (Wu, 1997), which are the cases of fully binarizable word alignments. Subsequently we provide a breakdown of word alignments into subclasses of increasing complexity.

Word alignments are considered the initial point for extracting translation units. In our analysis we differentiate between two cases of translation units (TUs) that can be extracted from a word alignment:

Contiguous TUs Only phrase pairs (called Contiguous translation units TUs)

Discontiguous TUs All contiguous + discontiguous TUs

When quantifying the coverage of ITG for word alignments, we explicitly make a difference between these two cases. As expected, it is usually more difficult to provide ITG derivations for discontiguous TUs in word alignments than for contiguous TUs. We define *ITG coverage* as the percentage of word alignments that can be covered by some ITG. A word alignment is covered by ITG if and only if it is fully binarizable, i.e., all nodes in the HATs are binary branching. This is a specific class of HATs called Binarizable Inversion Transduction Trees (BITTs).

Besides BITTs, word alignments can be grouped into subclasses of HATs according to the complexity of the reordering operators – set-permutations – on the nodes in these HATs. Apart from the BITT case (derivable by ITGs), we also define two more subclasses:

---

enumerating all of them from the Hypergraph and writing their tree structures to a textfile, then reading this unpacked forest from the textfile by the tree visualization component. As this can become somewhat slow in case of many HATs, rendering rendering all HATs can be turned on in the GUI, but only showing the first one is used as the default option.

PETs Percentage of word alignments covered by different Permutation Trees (PETs) that are beyond BITTs, i.e., these are bijective word alignments that are non-binarizable,

HATs All remaining HATs that are beyond PETs, i.e., non-bijective cases consisting of many-to-many word alignments and possibly discontiguous TUs.

3.1. Word alignments covered by ITG: BITTs

Data Sets We use manually and automatically aligned corpora. Manually aligned corpora come from two datasets. The first (Graça et al., 2008) consists of six language pairs: Portuguese–English, Portuguese–French, Portuguese–Spanish, English–Spanish, English–French and French–Spanish. These datasets contain 100 sentence pairs each and distinguish *Sure* and *Possible* alignments. Following Søgaard and Kuhn (2009), we treat these two equally. The second manually aligned dataset (Padó and Lapata, 2006) contains 987 sentence pairs from the English-German part of Europarl annotated using the Blinker guidelines (Melamed, 1998). The automatically aligned data comes from Europarl (Koehn, 2005) in three language pairs (English–Dutch, English–French and English–German). The corpora are automatically aligned using GIZA++ (Och and Ney, 2003) in combination with the grow-diag-final-and heuristic. With sentence length cutoff 40 on both sides these contain respectively 945k, 949k and 995k sentence pairs.

ITG Coverage is defined as the percentage word alignments (sentence pairs) in a parallel corpus that can be covered by an instance of ITG. Clearly, coverage depends on the chosen semantic interpretation of word alignments: contiguous translation units (phrase pairs) or discontiguous translation units.[2]

Results Table 1 shows the coverage of ITG for the different corpora dependent on the two alternative definitions of *translation equivalence*. The first thing to notice is that there is just a small difference between the Grammatical Coverage scores for these two definitions. The difference is in the order of a few percentage points, the largest difference is seen for Portuguese–French (79% v.s 74% Grammatical Coverage), for some language pairs there is no difference. For the automatically aligned corpora the absolute difference is on average about 2%. We attribute this to the fact that there are only very few discontiguous TUs that can be covered by ITG in this data.

[2] A note here on the computation of coverage for the different subclasses of HATs such as ITG Coverage in case of BITTs. As mentioned before every alignment typically yields a set containing multiple alternative HATs, corresponding to the different possible minimal factorizations of phrase pairs. Each of these HATs in the set however is by itself sufficient to determine the complexity for the whole set, since it always holds that all HATs in the set have the same complexity. This property is hence used for the efficient computation of the coverage statistics, based on just the first HAT from the computed Hypergraph of HATs for an alignment.

Alignments Set	Coverage contiguous	Coverage discontiguous
Hand aligned corpora		
English–French	76.0	75.0
English–Portuguese	78.0	78.0
English–Spanish	83.0	83.0
Portuguese–French	78.0	74.0
Portuguese–Spanish	91.0	91.0
Spanish–French	79.0	74.0
LREC Corpora Average	80.8±5.5	79.2±6.7
English–German	45.4	45.3
Automatically aligned Corpora		
English–Dutch	45.5	43.6
English–French	52.8	50.0
English–German	45.6	43.7
Automatically aligned corpora average	48.0±4.20	45.8±3.6

Table 1: The ITG coverage for different corpora dependent on the interpretation of word alignments: contiguous only or including discontiguous translation units

The second thing to notice is that the scores are much higher for the corpora from the LREC dataset than they are for the manually aligned English–German corpus. The approximately double source and target length of the manually aligned English–German corpus, in combination with somewhat less dense alignments makes this corpus much harder than the LREC corpora. Intuitively, one would expect that more alignment links make alignments more complicated. This turns out to not always be the case. Further inspection of the LREC alignments also shows that these alignments often consist of parts that are *completely linked*. Such completely linked parts are by definition treated as atomic TUs, which could make the alignments look simpler. This contrasts with the situation in the manually aligned English–German corpus where on average less alignment links exist per word.

When we look at the results for the automatically aligned corpora at the lowest rows in the table, we see that these are comparable to the results for the manually aligned English–German corpus (and much lower than the results for the LREC corpora). This could be explained by the fact that the manually aligned English–German is not only Europarl data, but possibly also because the manual alignments themselves were obtained by initialization with the GIZA++ alignments. In any case, the manually and automatically acquired alignments for this data are not too different from the perspective of ITG. Further differences might exist if we would employ another class of grammars, e.g., full SCFGs.

One the one hand, we find that manual alignments are well but not fully covered by ITG. On the other, the automatic alignments are not covered well but ITG. This suggests that these automatic alignments are difficult to cover by ITG, and the reason

Kind of HATs	English-Dutch	English-French	English-German
BITTs (Binarizable permut.)	45.5%	52.8%	45.6%
PETs (Permut.)	52.6%	56.6%	52.6%
HATs (Set-permut.)	100.0%	100.0%	100.0%

Table 2: The ratio of the different subsets of HATs in the corpus: BITTs, PETs and HATs

could be that these alignments are built heuristically by trading precision for recall (cf. Och and Ney, 2003). Søgaard (2010) reports that full ITG provides a few percentage points gains over ITG.

Overall, we find that our results for the LREC data are far higher than Søgaard's results but lower than the upperbounds of Søgaard and Wu (2009). A similar observation holds for the English–German manually aligned EuroParl data, albeit the maximum length (15) used in (Søgaard and Wu, 2009; Søgaard, 2010) is different from ours (40). We attribute the difference between our results and Sogaard's approach to our choice to adopt lexical productions of ITG that contain own internal alignments (the detailed version) and determined by the atomic TUs of the word alignment. Our results differ substantially from Søgaard and Wu (2009) who report upperbounds (indeed our results still fall within these upperbounds for the LREC data).

3.2. Breakdown according to reordering complexity

Table 2 shows the breakdown statistics of word alignments according to complexity class for three of the automatically aligned EuroParl corpora. The first line with percentages for BITTs corresponds to the "Coverage Contiguous" scores for the automatically aligned corpora, at the bottom lines in the middle column in Table 1. There is an obvious difference between the percentage of BITTs for English-French vs. English-German and English-Dutch. This difference carries over to PETs, fully bijective word alignments. The percentages for PETs clearly show that approximately half of the word alignments in the data are beyond the bijective case.

4. Related Work

General tools have been created for the visualization of basic word alignment (Smith and Jahr, 2000; Germann, 2008) as well as for the manual annotation of sentence pairs. For the alignment of syntactic trees another available toolkit is the Stockholm Tree Aligner (Volk et al., 2007). As a somewhat related problem, Maillette de Buy Wenniger et al. (2010) take on visualization of the coherence of tree-based reordering with word alignments.

Normalized Decomposition Trees (NDTs) were introduced in Zhang et al. (2008), giving an efficient tree-based hierarchical representation of nested phrase pairs. Combined with these structures a very efficient phrase extraction algorithm with

linear time complexity was proposed. In these representations, the relative reordering operations taking place at the nodes is left implicit, as it is not relevant from the point of view of plain phrase extraction. Indeed for phrase extraction NDTs are a superior tool. On the other hand, for effective visualization and quick insight into the structure, explicit reordering labels play an important role. For efficient and intuitive reordering complexity analysis this is equally true. Hierarchical Alignment Trees can be seen as an extension of Normalized Decomposition Trees, that makes the reordering relations between translation equivalents explicit and also keeps track of the alignment relations within phrase pairs. The latter guarantees that for every word alignment there is a mapping to a set of HATs, such that the set of HATs truly captures all information and structure of the word alignment. This is not completely the case for NDTs.

The array of work described in Zens and Ney (2003); Wellington et al. (2006); Søgaard and Wu (2009); Søgaard and Kuhn (2009); Søgaard (2010) concentrates on methods for calculating *upperbounds* on the alignment coverage for all ITGs, including NF-ITG. See Maillette de Buy Wenniger and Sima'an (2013) for a more complete overview. Some of this work also uses alignment parsing to compute more exact scores (Søgaard, 2010), recently extended by Kaeshammer (2013) which adds new flavor to this discussion by looking at alignment reachability for Synchronous Linear Context-Free Rewriting Systems.

5. Conclusions

We introduced a toolkit for the visualization, search and analysis of hierarchical translation equivalence. We have shown how this toolkit can help to get a better qualitative as well as quantitative understanding of translation equivalence relations as induced by word alignments for real big translation data. The software is distributed under the LGPL license and can be downloaded from:

`https://bitbucket.org/teamwildtreechase/hatparsing`

Acknowledgements

This work is supported by The Netherlands Organization for Scientific Research (NWO) under grant nr. 612.066.929. The authors would like to thank Remko Scha for his valuable feedback on the manuscript. The authors would furthermore like to thank the anonymous reviewers for all their helpful and detailed suggestions and comments.

Bibliography

Germann, U. Yawat: yet another word alignment tool. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL-HLT)*, pages 20–23, 2008.

Graça, João, Joana Pardal, Luísa Coheur, and Diamantino Caseiro. Building a golden collection of parallel multi-language word alignment. In *LREC'08*, Marrakech, Morocco, 2008. European Language Resources Association (ELRA).

Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595, 2009.

Kaeshammer, Miriam. Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, 2013. URL `http://www.aclweb.org/anthology/W13-0808`.

Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*, 2005.

Maillette de Buy Wenniger, Gideon and Khalil Sima'an. A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the itg hypothesis. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 58–67, 2013. URL `http://www.aclweb.org/anthology/W13-0807`.

Maillette de Buy Wenniger, Gideon, Maxim Khalilov, and Khalil Sima'an. A toolkit for visualizing the coherence of tree-based reordering with word-alignments. *The Prague Bulletin of Mathematical Linguistics*, pages 97–106, 2010.

Melamed, Dan. Annotation style guide for the blinker project, version 1.0. Technical Report IRCS TR #98-06, University of Pennsylvania, 1998.

Och, Franz Josef and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Padó, Sebastian and Mirella Lapata. Optimal constituent alignment with edge covers for semantic projection. In *ACL-COLING'06*, ACL-44, pages 1161–1168, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Satta, Giorgio and Enoch Peserico. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810, 2005.

Sima'an, Khalil and Gideon Maillette de Buy Wenniger. Hierarchical alignment trees: A recursive factorization of reordering in word alignments with empirical results. ILLC Prepublication series: Technical Report PP-2011-38, 2013. URL `http://staff.science.uva.nl/~simaan/D-Papers/HATsReport2013.pdf`.

Smith, Noah A. and Michael E. Jahr. Cairo: An alignment visualization tool. In *Proc. of the 2nd Conference on Language Resources and Evaluation (LREC)*, page 549–551, 2000.

Søgaard, Anders. Can inversion transduction grammars generate hand alignments? In *Proccedings of the 14th Annual Conference of the European Association for Machine Translation (EAMT).*, 2010.

Søgaard, Anders and Jonas Kuhn. Empirical lower bounds on alignment error rates in syntax-based machine translation. In *SSST '09*, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Søgaard, Anders and Dekai Wu. Empirical lower bounds on translation unit error rate for the full class of inversion transduction grammars. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 33–36, 2009.

Volk, M., J. Lundborg, and M. Mettler. Alignment tools for parallel treebanks. In *In Proc. of The Linguistic Annotation Workshop at the Association for Computational Linguistics (LAW-ACL)*, 2007.

Wellington, Benjamin, Sonjia Waxmonsky, and I. Dan Melamed. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Wu, Dekai. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 3(23):377–403, 1997.

Zens, Richard and Hermann Ney. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the Annual Meeting of the ACL*, pages 144–151, 2003.

Zhang, Hao, Daniel Gildea, and David Chiang. Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, pages 1081–1088, 2008.

Address for correspondence:
Gideon Maillette de Buy Wenniger
gemdbw AT gmail.com
Institute for Logic Language and Computation
Faculty of Science,
University of Amsterdam
Science Park 107
1098 XG Amsterdam, Netherlands

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 55–69

Pipeline Creation Language for Machine Translation

Ian Johnson

Capita Translation and Interpreting

Abstract

A pipeline is a commonly used architecture in machine translation (MT). Statistical MT can require, for example, many computational steps during training and decoding which are best viewed as a pipeline. These pipelines can be tedious to construct and are error prone since little or no type checking is performed, and no clear interface is defined for a pipeline's components. At times this can manifest itself as components requiring knowledge of how a preceding component's output is formed in order to consume it. Moreover, collaboration and sharing of pipelines or components becomes difficult since the components themselves may need to be changed in order to be used by others.

In order to alleviate these problems a specialised language has been designed called Pipeline Creation Language (PCL). PCL allows users to construct pipelines that have components with well defined and compatibility checked interfaces. Components can be defined in packages, so individual components or packages of components, including entire pipelines, can be shared and composed with others. PCL supports operators which allow components to be executed sequentially, in parallel, or conditionally.

1. Introduction

Machine translation seems to attract constructions that are best seen as pipelines. For example, statistical MT (SMT) tends to use pipelines to build up the computation needed for its training and decoding phases. Pipelines typically contain many components that can:

- Have no clear interface: What types of input data are required for this component to compute which types of output data? Also, what types of data are required to instantiate the component? Or,

© 2014 PBML. All rights reserved.
Corresponding author: ian.johnson@capita-ti.com
Cite as: Ian Johnson. Pipeline Creation Language for Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 55–69. doi: 10.2478/pralin-2014-0004.

- Be too coupled: A subsequent component may need to alter the data on its input from another component. The logic needed to do this is usually placed in the subsequent component. This "glue" logic prevents the component being easily re-used in the same or another pipeline.

In this case the user of a component does not know whether it is compatible with others. Is the data, or the format of the data being output compatible with another component a developer wishes to use? Plus, these problems can prevent components being shared by a community and so limiting collaboration.

The development and maintenance of pipelines can be difficult and tedious. Typical pipeline implementations, with many components, can bear little resemblance to their conceptual view. Moreover, the implementations tend to be monolithic which can lead to developers becoming lost in a mass of code. Due to these difficulties coding errors can occur that are missed.

Pipeline Creation Language (PCL) provides a specialised language that allows the construction of non-recurrent software pipelines that provides:

- **Compatibility checking**: Components define an interface that specifies named data flowing to and from input and output ports. The ports are checked so that the compatibility of two connecting components can be determined.
- **Packages and modules**: Components can be defined in a hierarchical manner. This eases the management of pipelines with many components since one component can be worked on at once.
- **Testing**: Components can be easily tested in isolation using the PCL runtime and configuration files.
- **Promotes component re-use**: Components should be developed in isolation from any other component such that they could be reused. Also, libraries of components can be constructed using packages. If "glue" logic is needed to attach two components then the "knowledge" of how to do this resides in the recipients pipeline.
- **Representational implementation**: Pipeline implementations "look" like pipelines, unlike when implemented using a procedural approach.
- **Sharing and collaboration**: All of the above make sharing of components and collaboration easier. This is how modern programming languages create ecosystems of libraries that are sharable and used in many applications.

A PCL compiler and runtime, including documentation, is freely available, under a LGPL v3 license, from GitHub by cloning `https://github.com/ianj-als/pcl.git`.

2. PCL and Experiment Management

Experiment managers allow experimenters to run and re-run experiments using a collection of scripts or executables configured into, usually, a software pipeline and handle experiment inputs and experiment results. An experiment manager ensures that each run of an experiment has a distinct location for its results. Experiment man-

agers, such as Moses' EMS (Philipp Koehn, 2010) and Eman (Ondřej Bojar and Aleš Tamchyna, 2013), handle two concerns; the definition of a software pipeline, and collating results.

The PCL language can be used for the software pipelining aspect of an experiment manager. The PCL runtime could be used to build an experiment manager since the runtime presents a public API. For more information on the runtime API please see the PCL user manual (Ian Johnson, 2013).

3. Implementing an SMT Training Pipeline

In order to demonstrate how PCL can be useful for MT pipelines, a simplified training pipeline, shown in Figure 1, is to be constructed. The training pipeline generates a translation and language model, and then tuning is done to produce models which can be used in a decoding pipeline. A similar pipeline, along with documentation, can be found in the Moses GitHub repository, see `contrib/arrow-pipelines` in a clone of `https://github.com/moses-smt/mosesdecoder.git`.

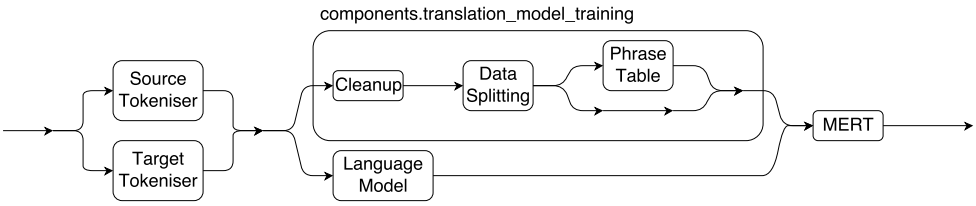

Figure 1. Simple SMT training pipeline.

3.1. Pipeline Components

In PCL a pipeline component is a reusable unit of computation that is instantiated and used in other components. Existing programs that are being used in your pipelines can be adapted for use by wrapping them in a PCL module. This PCL component can be imported, and combined with other PCL components.

PCL components can have 2, 3, or 4 *ports*, and each port should contain one or more *signals*. A port is the point, or points, at which one component attaches to another. Components may attach to each other if, and only if, they have a matching number of output and input ports. A signal is a piece of named and "duck typed" data that flows through a port. Ports can have an unlimited number of signals. *Port specifications* are used to determine the number of input and output ports and their signals. Single ports are defined as a comma separated list of signal names, e.g. `corpus.source.filename, corpus.target.filename`. Dual ports are defined using two parenthe-

sised and comma separated list of signal names, e.g. `(corpus.source.filename, corpus.source.no_sentences), (corpus.target.filename, corpus.target.no_sentences)`. This information is used to determine whether components are compatible with each other.

There are two kinds of component in PCL and use slightly different syntax, they are:

- **Computational Component**: Represents a computation that shall be evaluated when the component is executed. These components only have dependencies on the PCL runtime and are "leaves" in the component dependency tree.
- **Combinator Component**: These components combine other components, both computational and combinator kinds, to create new components.

Figure 2 shows the component model used in PCL. This is a structural design pattern called the *composite* pattern (Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, 1994).

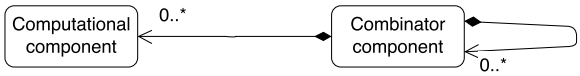

Figure 2. PCL's component model.

3.2. Computational Component

An example of how a batch tokeniser can be wrapped in PCL is shown in Figure 3. Those familiar with Haskell will see that the imperative style is similar to *do-notation* (see Miran Lipovača (2011)).

This PCL file defines the computation for a tokeniser component. It is composed of the following sections:

- **Imports** (lines 1-5): These imports are part of the runtime library and provide basic and common operations. These files are written in Python and define functions. Once imported the module alias is used with the function name to generate a function call, e.g., `path.join(...)`. The search for runtime library imports is determined by the `PCL_IMPORT_PATH` environment variable, and allows users to define their own runtime libraries.
- **Name** (line 7): A meaningful name for the component. Components should be uniquely named in a package. The file name of the component must be the same as the component's name, e.g., a file called `tokeniser.pcl` must contain a component called `tokeniser`.
- **Ports** (lines 8 & 9): The input and output port specifications of the component. Computational components can *only* define one input and one output port.

```
1  import pcl.io.file as file
2  import pcl.os.path as path
3  import pcl.system.process as process
4  import pcl.util.list as list
5  import pcl.util.string as string
6
7  component tokeniser
8    input corpus.filename
9    output corpus.tokenised.filename
10   configuration corpus.language, working.directory.root, moses.installation
11   do
12     language <- string.lower(@corpus.language)
13
14     corpus.file.basename <- path.basename(corpus.filename)
15     corpus.file.basename.bits <- string.split(corpus.file.basename, ".")
16     list.insert(corpus.file.basename.bits, -1, "tok")
17     result.basename <- string.join(corpus.file.basename.bits, ".")
18     result.pathname <- path.join(@working.directory.root, result.basename)
19
20     working.exists <- path.exists(@working.directory.root)
21     if working.exists == False then
22       path.makedirs(@working.directory.root)
23       return ()
24     else
25       return ()
26     endif
27
28     tokeniser.cmd <- path.join(@moses.installation, "scripts",
29                                "tokenizer", "tokenizer.perl")
30     tokeniser.cmd.line <- list.cons(tokeniser.cmd, "-l", language, "-q")
31
32     corpus.file <- file.openFile(corpus.filename, "r")
33     result.file <- file.openFile(result.pathname, "w")
34     process.callAndCheck(tokeniser.cmd.line, corpus.file, result.file)
35     file.closeFile(result.file)
36     file.closeFile(corpus.file)
37
38     return corpus.tokenised.filename <- result.pathname
```

Figure 3. `tokeniser.pcl`: An example of how an existing tokenisation script can be used in PCL.

- **Configuration** (line 10): The configuration is static data which is used to instantiate the component. This data is optional since components may not require any parameters to construct them. Prefixing an identifier with `@` references a configuration value, e.g., `@corpus.language`.
- **Computation** (lines 11-38): The component must define the computation that maps input signals to output signals. This section begins with the `do` keyword. Execution flows from top to bottom and each line yields a value which can be assigned to a "write once" identifier. This section *must* end with a return statement which assigns values to all output signals.

Once all of the computational components have been defined they can be combined, using PCL, to create more complex components. The next section describes how PCL combines components into, in this instance, the training pipeline.

3.3. Pipeline Implementation

A combinator component representing the training pipeline is shown in Figure 12. A PCL file that defines a combinator component is composed of the following sections:

- **Imports** (lines 1-4): Imports can be optionally specified. Importing, as in other languages, makes available other components to the PCL component being written. PCL components can be defined in namespaces. PCL components in namespaces must be fully qualified by using dot separated names. The search for PCL components is determined by the `PCL_IMPORT_PATH` environment variable. This is a colon separated list of directories where PCL namespaces are to be found. Only one PCL component can be imported with one import statement and each imported component must be given an alias. The component shall be known by its alias.
- **Component** (line 6): This starts the component definition and provides its name. The file name of the component must be the same as the component's name. For example, a component defined in `training_pipeline.pcl` must be called `training_pipeline`.
- **Inputs and Outputs** (lines 7 and 8): Defines the input and output port specifications of the component. This information is used to determine if a component is compatible with another.
- **Configuration** (lines 9-13): This is an optional section that defines static data which shall be used to construct this component.
- **Declarations** (lines 14-36): This optional section is used to construct components which will have been imported. The imported component's alias is used to build, possibly, more than one instance of the component. If an imported component requires configuration to be constructed the importing component's configuration must be mapped using a declaration's *with* clause. The with clause defines which configuration, possibly renamed, shall be passed to the component's constructor.
- **Definition** (lines 37-58): Beginning with the `as` keyword, this section defines a single expression which represents the pipeline. Constructed components, predefined components (see Section 3.5) and combinator operators (see Section 3.4) can all be used to create the component's definition.

3.4. Component Combinator Operators

Behind the scenes, combinator components are implemented as *arrows* (John Hughes (2000), Paterson (2001), Paterson (2003), John Hughes (2005), and Conor McBride

(2011)). Arrows are abstractions of computation that define a set of combinator operators which construct, as a result, another arrow. Arrows can be combined indeterminately and this allows any arbitrarily complex PCL component to be used in another component. There are five component combinator operators defined in PCL, they are:

- **Composition**: Join one component's output to the input of another component, e.g., the PCL expression composes components `f : b → c` and `g : c → d` with `f >>> g`. The PCL compiler shall compatibility check the two components when using the composition combinator. The input and output port specifications are `b` and `d` respectively.

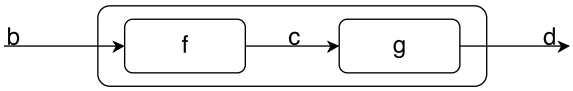

Figure 4. >>>: Component composition.

- **First**: A component that takes a component, e.g. `f : b → c`, which will apply the first element of the input pair to `f`. The second element of the pair passes through unchanged. In PCL, `first f`. The input and output port specifications are `(b), (d)` and `(c), (d)` respectively.

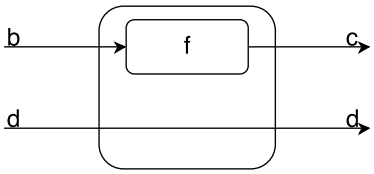

Figure 5. `first`: First element of input pair applied to component `f`.

- **Second**: Similar to first only the second element of the input pair is applied to the component, and the first element passes through unchanged. In PCL, `second f`. The input and output port specifications are `(d), (b)` and `(d), (c)` respectively.
- **Parallel**: The components `f : b → c` and `g : d → e` can be executed in parallel by using `***` combinator. In PCL, `f *** g`. The input and output port specifications are `(b), (d)` and `(c), (e)` respectively.
- **Fanout**: The components `f : b → c` and `g : b → d` receive the same input and are executed in parallel. In PCL, `f &&& g`. The input and output port specifications are `b` and `(c), (d)` respectively.

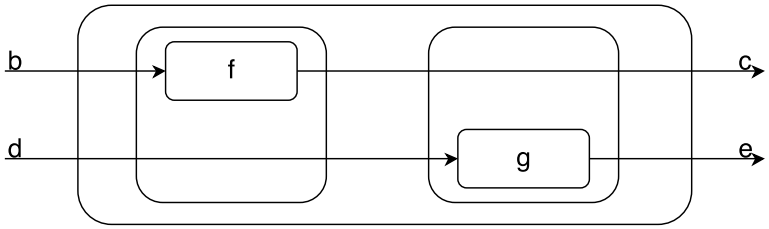

Figure 6. ∗∗∗: Parallel components.

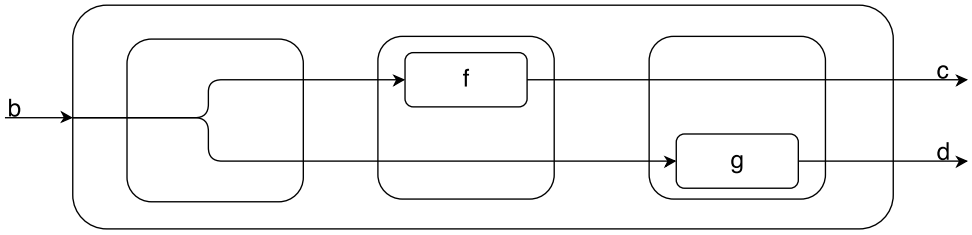

Figure 7. &&&: Input applied to parallel components.

3.5. Pre-defined Components

There are four pre-defined components available for use in PCL which split, merge and convert signals. Also, there is an *If* component which allows components to be conditionally executed.

3.5.1. Split

The `split` component is a component with a single port and two output ports. This component takes the signal(s) on the input port and copies them to each of the output ports.

3.5.2. Merge

The `merge` component has a dual input port and a single output port. This component merges signals from the input pair to unique signals in the output port. Signals in the input ports are used as indices to the keywords `top` and `bottom` referring the the top and bottom input port. Also, literal values can be "injected" in a merge component or signals can be dropped. All top and bottom signals must be specified in a merge. The example merge component in Figure 8 shows the top port's signals

`eval.data.filename`, and `eval.data.size` begin mapped to the output signal `evaluation_filename` and dropped respectively. Also, the `language.model.filename` sig-

```
merge top[eval.data.filename] -> evaluation_filename,
      top[eval.data.size] -> _,
      bottom[language.model.filename] -> language_model,
      9 -> language_model_type
```

Figure 8. An example merge component.

nal from the bottom port is mapped to output signal `language_model`, and the literal value `9` is mapped to an output signal called `language_model_type`. This merge component has the following input and output port specifications

`(eval.data.filename, eval.data.size), (language.model.filename)`

and

`evaluation_filename, language_model, language_model_type`

respectively.

3.5.3. Wire

A wire component renames signals so that components can be used together. Wires can have 2 ports, one input and output port, or 4 ports, two input and output ports. Two port wires contain one signal mapping such as shown in Figure 9, and a four port wire is shown in Figure 10. As with merge components, literal values can be injected and signals can be dropped in wires, but all input signals must be specified.

```
wire src_filename -> filename,
     src_language -> language,
     src_no_sentences -> _,
     True -> is_file_clean
```

Figure 9. An example two port wire component.

```
wire (tokenised_filename -> tokenised_src_filename,
      tokenised_filename_size -> _,
      "de" -> tokenised_src_language),
     (tokenised_filename -> tokenised_trg_filename,
      tokenised_filename_size -> _,
      "en" -> tokenised_trg_language)
```

Figure 10. An example four port wire component.

3.5.4. Conditional Execution with *If*

Components can be conditionally executed using the *If* component. The *If* component takes three arguments:

- **Condition expression**: when evaluated if this is a "truthy" value the *then* component is executed, otherwise the *else* component is executed.
- **Then component**: a component which is executed on the condition being of a "truthy" value, and
- **Else component**: a component which is executed on the condition *not* being of a "truthy" value.

Both the *then* and *else* components must have identical input and output port specifications. The condition can contain the usual comparison operators (`==`, `!=`, `>`, `<`, `>=`, and `<=`), logically operators (`or`, `and` and `xor`), input port signal names and literal values. For example, `src_language == "en" and trg_language != "th"`.

4. PCL Compiler

The PCL compiler is located in `src/pclc` of your Git clone, and is called `pclc.py`. The compiler has a number of command line options that are shown in Table 1 along with their meanings.

Option	Meaning
`-i, --instrument`	The object code is instrumented with logging messages. These messages will appear on `stderr` when the pipeline is executed.
`-l LOGLEVEL, --loglevel=LOGLEVEL`	Logging level of the compiler during compilation. This affects the content of the `pclc.log` file. Valid values of `LOGLEVEL` are: `CRITICAL`, `ERROR`, `WARNING`, `WARN`, `INFO`, `DEBUG`.
`-v, --version`	Show the compiler's version and exits.
`-h, --help`	Shows the compiler's help information and exits.

Table 1. PCL compiler command line options.

To compile a PCL component called `tokeniser.pcl` use the command:

```
pclc.py tokeniser.pcl
```

This shall generate three files: the object file `tokeniser.py`, a `__init__.py` file, and a compilation log file called `pclc.log`.

5. PCL Runtime

The PCL runtime can be found in the `src/pcl-run` directory of your Git clone. The PCL runtime has some command line options that are shown in Table 2 along with their meanings.

Option	Meaning
`-n NO_WORKERS, --noworkers=NO_WORKERS`	Number of pipeline evaluation threads and defaults to 5 threads. The runtime executes the pipeline in a thread pool whose size is governed with this option. The performance of a pipeline may be dependent on the value used.
`-v, --version`	Show the compiler's version and exits.
`-h, --help`	Shows the compiler's help information and exits.

Table 2. PCL runtime command line options.

The runtime requires a configuration file, specified on the command line, to run the component of the same name. To execute a component called `tokeniser` invoke the runtime using:

```
pcl-run.py tokeniser.cfg
```

On `stdout`, providing all goes well, the output of the pipeline shall be displayed. The `PCL_IMPORT_PATH` environment variable is used to search for runtime libraries and compiled components.

5.1. Runtime Configuration File

The pipeline configuration file contains the static configuration and the pipeline's inputs. The configuration filename must be the same as the component you wish to run with a `.cfg` extension, e.g. the `tokeniser` configuration file must be called `tokeniser.cfg` and must be in the same directory. The configuration file contains two sections `[Configuration]`, for configuration values, and `[Inputs]`, for pipeline inputs. Each section contains key value pairs, e.g. the `tokeniser` configuration might look like the example in Figure 11. Environment variables can be used in configuration files with `$(VAR_NAME)`. The environment variable, if it exists, shall be substituted and used in the pipeline. The ability to execute any pipeline component facilitates testing during pipeline development. It is recommended that a configuration file and test data be developed alongside a component.

```
1  [Configuration]
2  corpus.language = en
3  working.directory.root = tokenisation
4  moses.installation = /opt/moses
5  [Inputs]
6  corpus.filename = my_corpus.src
```

Figure 11. An example PCL configuration file.

6. PCL Performance

The pipeline implemented here is shown in Figure 1. It is used to show the runtime overhead using PCL compared with a Bash script implementation. The PCL implementation is shown in Figure 12 and is the *top* level component definition which defines the entire pipeline. The Bash script, however, implements each of the components to emulate the PCL pipeline, and executes each "component" sequentially. The entire PCL implementation and the Bash script can be found in the `contrib/arrow-pipelines` directory of the Moses GitHub repository.[1]

The parallel corpus used is 50,000 sentences extracted from the English to Dutch Europarl corpus. Each training pipeline implementation and thread count combination was executed three times. The PCL implementation executed with 1 thread is equivalent to the Bash script, when executed, since components in both implementations are forced to execute sequentially.

The execution times for each performance experiment is shown in Table 3. The times shown represent the median, minimum and maximum execution times, in seconds, for each performance experiment.

Implementation	Thread Count	Real execution times (s)		
		Median	Min	Max
PCL pipeline	5	276	275	278
PCL pipeline	1	279	277	281
Bash script	1	650	643	654

Table 3. Performance results for the implemented training pipeline.

The PCL implementation completes in under half the time of the Bash script implementation. This is due a slow running implementation of the *Cleanup* component in the Bash implementation of the pipeline. The Bash implementation takes around 390s to execute, compared to around 1s in Python. The Bash implementation of the *Cleanup* component uses the command `wc` to count the number of words in each source and

[1] `https://github.com/moses-smt/mosesdecoder.git`

target sentence in the corpus. The *wc* command is, therefore, executed 100,000 times for this corpus. This means that at least 100,000 processes, but is probably as many as 500,000 processes, need to be spawned in order to process this corpus. The *wc* command, once cached, will execute in around 1ms; including the other processes required in this component it is conceivable that the elapsed time, on the test hardware, could be as much as 500s.

Increasing the number of threads for this pipeline only produces a marginal performance increase since the number of active branches in this pipeline is low. Moreover, the *Mert* component can only execute once *all* other components have completed and generated their output signals, and any files that need to be written to disk. It, also, takes up much of the execution time, around 180s in both implementations.

Discounting the *Cleanup* component's performance in Bash, the PCL implementations take around 7% longer to execute than the Bash script using this parallel corpus. The PCL runtime, therefore, introduces a minimal execution time overhead, and the pipeline developer gets all the advantages of the PCL language and component validations by the compiler.

7. Summary

Pipeline creation language (PCL) is a specialised, and modular language for building non-recurrent software pipelines. This paper briefly describes how PCL can be used to construct an, albeit simplified, SMT training pipeline. The PCL language, and how to integrate existing programs with PCL was described along with a performance test to show the minimal impact on execution time. It is recommended that the PCL user manual (Ian Johnson, 2013) be consulted for further details.

Acknowledgements

This work was done as part of the MosesCore project sponsored by the European Commission's Seventh Framework Programme (Grant Number 288487).

```
import components.translation_model_training as model_training
import components.wrappers.irstlm_build.irstlm_build as lang_model
import components.wrappers.mert.mert as mert
import components.wrappers.tokeniser.tokeniser as tokeniser

component training_pipeline
  inputs src_filename, trg_filename
  output moses_ini_filename
  configuration source_language, target_language, max_segment_length, corpus_development_size,
                corpus_evaluation_size, alignment_method, reordering_method, smoothing_method,
                tokenisation_directory, translation_model_directory, language_model_directory,
                mert_directory, moses_installation_directory, giza_installation_directory,
                irstlm_installation_directory
  declare
    src_tokeniser := new tokeniser with source_language -> corpus.language,
                                        tokenisation_directory -> working.directory.root,
                                        moses_installation_directory -> moses.installation
    trg_tokeniser := new tokeniser with target_language -> corpus.language,
                                        tokenisation_directory -> working.directory.root,
                                        moses_installation_directory -> moses.installation
    model_training := new model_training with max_segment_length -> model_training.max_segment_length,
                          corpus_development_size -> model_training.corpus.development_size,
                          corpus_evaluation_size -> model_training.corpus.evaluation_size,
                          translation_model_directory -> model_training.translation_model.dir,
                          alignment_method -> model_training.method.alignment,
                          reordering_method -> model_training.method.reordering,
                          source_language -> model_training.src.language,
                          moses_installation_directory -> model_training.moses.installation,
                          giza_installation_directory -> model_training.giza.installation,
                          target_language -> model_training.trg.language
    irstlm := new lang_model with irstlm_installation_directory -> irstlm_installation_dir,
                                  smoothing_method -> irstlm_smoothing_method,
                                  language_model_directory -> language_model_directory
    mert := new mert with source_language -> source_language, target_language -> target_language,
                          moses_installation_directory -> moses_installation_dir,
                          mert_directory -> mert_working_directory
  as
    (wire src_filename -> src_filename, trg_filename -> _ &&&
     wire trg_filename -> trg_filename, src_filename -> _) >>>

    (wire (src_filename -> corpus.filename), (trg_filename -> corpus.filename) >>>
     (src_tokeniser *** trg_tokeniser) >>>
     wire (corpus.tokenised.filename -> tokenised_src_filename),
          (corpus.tokenised.filename -> tokenised_trg_filename)) >>>

    merge top[tokenised_src_filename] -> tokenised_src_filename,
          bottom[tokenised_trg_filename] -> tokenised_trg_filename >>>

    ((wire tokenised_src_filename -> src_filename, tokenised_trg_filename -> trg_filename >>>
      model_training) &&&
     (wire tokenised_trg_filename -> input_filename, tokenised_src_filename -> _ >>> irstlm)) >>>

    merge top[moses_ini_filename] -> moses_ini_filename,
          top[evaluation_data_filename] -> evaluation_data_filename,
          bottom[compiled_lm_filename] -> trg_language_model_filename,
          bottom[add_start_end_filename] -> _, bottom[lm_filename] -> _,
          3 -> trg_language_model_order, 9 -> trg_language_model_type >>>
    mert
```

Figure 12. PCL implementation of the simple training pipeline.

Bibliography

Conor McBride. Kleisli arrows of outrageous fortune. (unpublished), 2011.

Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, chapter Structural Patterns. Addison-Wesley, 1994. ISBN 0201633612.

Ian Johnson. *Pipeline Creation Language (PCL): User Manual*. Capita Translation and Interpreting, 2013. URL `https://github.com/ianj-als/pcl/blob/master/documentation/pcl-manual.latest.pdf`.

John Hughes. Generalising Monads to Arrows. *Science of Computer Programming*, 37:67–111, May 2000.

John Hughes. Programming with Arrows. In *Advanced Functional Programming*, pages 73–129. Springer Berlin Heidelberg, 2005.

Miran Lipovača. *Learn You a Haskell for Great Good!*, chapter A Fistful of Monads. No Starch Press, 2011. ISBN 1593272839.

Ondřej Bojar and Aleš Tamchyna. The Design of Eman, an Experiment Manager. *The Prague Bulletin of Mathematical Linguistics*, 99:39–58, September 2013.

Paterson, Ross. A new notation for arrows. In *International Conference on Functional Programming*, pages 229–240. ACM Press, Sept. 2001.

Paterson, Ross. Arrows and computation. In *The Fun of Programming*, pages 201–222. Palgrave, 2003.

Philipp Koehn. An Experimental Management System. *The Prague Bulletin of Mathematical Linguistics*, 94:87–96, September 2010.

Address for correspondence:
Ian Johnson
`ian.johnson@capita-ti.com`
Capita Translation and Interpreting,
Riverside Court, Huddersfield Road,
Delph, Lancashire,
OL3 5FZ, United Kingdom.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 71–96

Czech Machine Translation in the project CzechMATE

Ondřej Bojar, Daniel Zeman

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

Abstract

We present various achievements in statistical machine translation from English, German, Spanish and French into Czech. We discuss specific properties of the individual source languages and describe techniques that exploit these properties and address language-specific errors. Besides the translation proper, we also present our contribution to error analysis.

1. Introduction

This overview article summarizes recent advances in machine translation involving Czech, as achieved within the project CzechMATE. The overall state of the art in machine translation has recently made a leap in quality, esp. with the introduction of phrase-based methods (Koehn, 2004; Koehn et al., 2007) and availability of very large corpora (Bojar et al., 2012). The output may still suffer many errors, including serious ones such as completely reversed meaning, but it is nevertheless rather difficult to improve it fully automatically. The main reason is that the spectrum of outstanding errors is diverse and the errors are not easy to fix without negatively affecting the rest of the sentence. Aware of this challenging situation, we experimented at multiple fronts, searching for the "lower-hanging fruit".

The article has two main parts: Section 2 is concerned with methods of machine translation evaluation, covering techniques that fully rely on human judgement, techniques fully automatic as well techniques and tools mixing the two. Section 3 describes a wide range of our experiments with statistical machine translation into Czech, be it small specific ideas such as handling of named entities or breaking words into morphemes, or a complex combination of three big components into the current best English-to-Czech MT system.

© 2014 PBML. All rights reserved.
Corresponding author: zeman@ufal.mff.cuni.cz
Cite as: Ondřej Bojar, Daniel Zeman. Czech Machine Translation in the project CzechMATE. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 71–96. doi: 10.2478/pralin-2014-0005.

2. Novel Methods of Machine Translation Evaluation

Measuring progress is a critical component of scientific work but evaluating machine translation is unfortunately a rather peculiar task.

When comparing two possible MT outputs relative to each other, be it two distinct MT systems or an older vs. a newer version of a single system as a progress check, the hypotheses are often incomparable. One sentence can have an error in the beginning, the other at the end. One can be more or less correct but disfluent, the other can be perfectly fluent but reverse the meaning.

When assessing the quality of a single hypothesis on an "absolute" scale, we face subjectivity of human judgments: if the output of the MT system is not error-free right away, it is usually not clear which part is wrong because an input sentence has many possible translations, see also Section 2.5. Also, some errors are more serious than others, e.g. some distortion of the meaning vs. a clear and non-confusing typo or a typographical error. Moreover, each annotator gets quickly accustomed to the errors of the system and it is increasigly hard for him to notice them.

The field of MT evaluation is thus actively evolving and no completely satisfactory solution has been found so far. In our project, we contributed to both manual (Sections 2.1, 2.2 and 2.3) and automatic evaluation methods (Sections 2.4 and 2.5).

2.1. Tools for Manual MT Output Inspection and Analysis

We developed an interactive tool for analysis of MT errors, called Addicter (Automatic Detection and DIsplay of Common Translation ERrors) (Berka et al., 2013). It can do the following:

- Find erroneous tokens and classify the errors in a way similar to Vilar's taxonomy (Vilar et al., 2006). This component for automatic error detection and classification is further discussed in Section 2.4.
- Browse the test data, sentence by sentence, and show aligned source sentence, reference translation and system hypothesis (Figure 1).
- Browse aligned training corpus and look for example words in context.
- Show lines of the phrase table that contain a given word.
- Summarize alignments of a given word. This feature can also serve as a primitive corpus-based dictionary.
- Search and group words sharing the same lemma. That way, morphological errors can be highlighted.

The test data browser facilitates examination of system-generated hypothesis and its comparison to the reference translation(s). On the other hand, the search engine that operates on training corpus and phrase table can reveal whether an out-of-vocabulary word really never occurred in training data, or it got filtered out during subsequent processing.

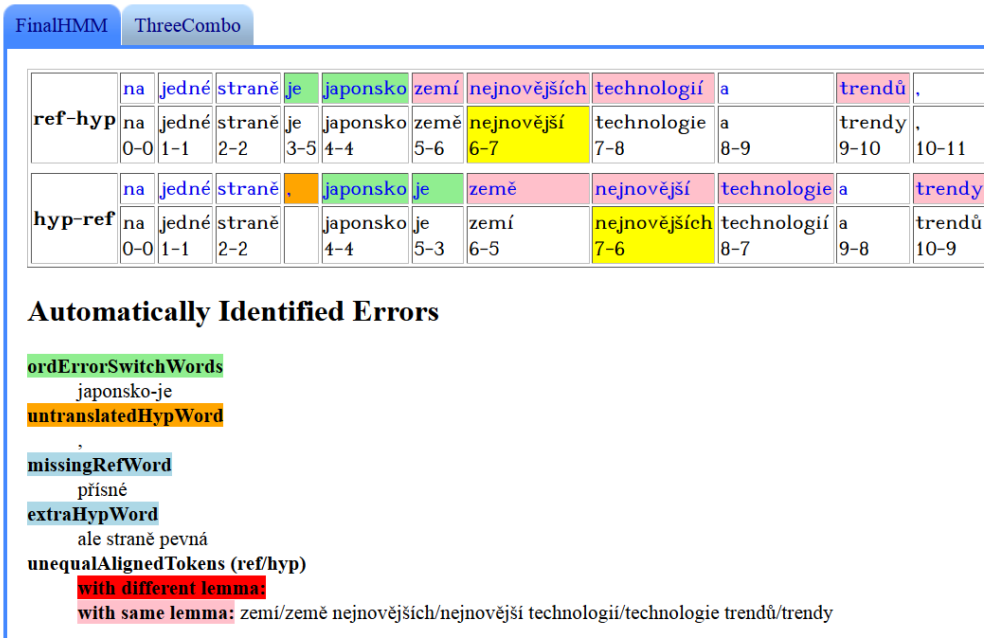

Figure 1. Screenshot of the Test Data Browser in Addicter. Different types of errors are highlighted using different colors. Monolingual word alignment between reference translation and system hypothesis is indicated using numerical indexes of words, and also highlighted when the mouse pointer is over a word. The English source of the sentence in the example is "On the one hand, Japan is the land of the latest technologies and trends, but on the other hand it is strict, disciplined and traditional."

2.2. Evaluating Predicate-Argument Structure

An established manual evaluation method asks annotators to RANK up to five different MT outputs for a given input sentence. The method is not sufficiently reliable (see also Section 3.7 below) but it has nevertheless been in use for a long series of WMT evaluation campaigns, see Callison-Burch et al. (2007) through Bojar et al. (2013a). The assignment for the ranking method is very simple: "You are shown a source sentence followed by several candidate translations. Your task is to rank the translations from best to worst (ties are allowed)." This simplicity is flexible with respect to overall system quality and allows to include untrained annotators in the evaluation but comes at a cost: people may focus on different aspects of the outputs, and even a single annotator may use inconsistent "quality scales" across sentences.

Reference	Oblečky	musíme	vystříhat	z časopisů
Gloss	clothes	we-must	cut	from magazines
Roles	Experiencer	Modal	Action	Locative
Meaning	We must cut the clothes (assuming paper toys) from magazines			
Hypothesis	Musíme	vyříznout	oblečení z časopisů	
Gloss	We-must	cut	clothes from magazines	
Roles	Modal	Action	Experiencer	

Figure 2. HMEANT: labelling semantic roles of phrases in the reference translation and the hypothesis. The example illustrates a problem of PP-attachment mismatch. One of the annotations treats the phrase "z časopisů" correctly as a separate frame filler, labelling it as Locative, the other annotation groups it together with the noun "oblečení" into an Experiencer. The subsequent alignment of slot fillers is not possible since two slots in the reference (Experiencer and Locative) correspond to one slot in the hypothesis (Experiencer).

Lo and Wu (2011) propose HMEANT, a manual method that evaluates MT output based on the core predicate-argument structure of the sentence. HMEANT checks, whether key elements of clause structure were preserved: "who did what to whom, how where and why". HMEANT was designed with future full automation in mind and its principles are very much in line with the structuralist linguistic theory (Sgall et al., 1986; Panevová, 1994; Lopatková et al., 2008) and processing tools (Popel and Žabokrtský, 2010) as developed at our institute. We thus decided to apply HMEANT to Czech, the first language other than English where HMEANT has been tested so far.

HMEANT consist of two phases:

Semantic role labelling where both the reference and the MT output receive an explicit annotation of which word is the predicate and which groups of words correspond to its arguments (or "semantic role fillers").

Alignment where predicates and their arguments in the hypothesis and in the reference get aligned whenever possible.

The final formula of HMEANT then evaluates the f-score of the match between the predicates and their arguments in aligned sentences.

Our experiment, as detailed in Bojar and Wu (2012), confirmed that annotators feel more confident when following the relatively simple HMEANT guidelines compared to the standard hypothesis ranking. On the other hand, we identified a number of issues with the current design of HMEANT and we proposed various changes to the guidelines. One example is depicted in Figure 2: the role labelling, done before alignment, sometimes produces annotations that cannot be aligned unless HMEANT allows for up to many-to-many alignments between arguments.

A subsequent independent experiment by Birch et al. (2013) observed similar problems for English and German and, more importantly, revealed that the inter-annotator agreements on individual labellings (the identification of the predicate, the filler spans as well as their semantic role labels) is relatively good but the alignment is problematic and reaches much lower agreement rates. Also, the discriminatory power of HMEANT is sometimes too weak, since only verbal frames are considered.

In conclusion, HMEANT is a step in a promising direction of MT evaluation but several issues need to be addressed.

2.3. Quiz-Based Evaluation

Most of non-comparative MT evaluation methods focus on whether the translation conveys the general meaning of the original and/or whether it satisfies various linguistic conditions in the target language. Some tests also check the general understandability of the sentence: "Tell us, what the sentence says", validated by a second annotator (Callison-Burch et al., 2009, 2010; Bojar et al., 2013a). In many practical situations, the output of MT can be nevertheless useful even if it contains many errors and omissions. We wanted to try also this much more applied type of evaluation.

Running a genuine task-based evaluation, e.g. bringing an object that matches a machine-translated description, is very expensive. We thus opted for a lighter variant: annotators were given short text snippets translated from native English to Czech by one of four evaluated MT systems. The task was to answer three Czech yes/no questions about the snippet.

Table 1, reproduced from our detailed description of the experiment (Berka et al., 2011), documents that different evaluation methods promote different systems. Specifically, the deep-syntactic system TectoMT in 2010 (see also Section 3.7) seemed to perform relatively poorly in terms of general understandability or n-gram based metrics like BLEU (Papineni et al., 2002) but it supported best our annotators in answering quiz questions.

It is worth noting that quiz-based evaluation scores are generally much higher than the other manual scores (all manual scores range from 0 to 100), which indicates that finding the answer to a specific question in machine-translated text is easier than understanding the whole sentences. Part of this result can be attributed to the fact that the question itself may help a lot in understanding the MT output and also that it may ask for just one or two words of the output. Quiz-based evaluation is thus perhaps insufficiently discerning but it brings some optimism towards the practical utility of machine translation.

2.4. Automatic Error Identification

Automatic error classification in Addicter is based on finding erroneous words in the translation output and assigning a corresponding error class to each of them. The

Metric	Google	CU-Bojar	PC Translator	TectoMT
$\geq$ others (WMT10 official)	**70.4**	65.6	62.1	60.1
$>$ others	49.1	45.0	**49.4**	44.1
General understandability [%]	**55**	40	43	34
Quiz-based evaluation [%]	80.3	75.9	80.0	**81.5**
BLEU	**0.16**	0.15	0.10	0.12
NIST	**5.46**	5.30	4.44	5.10

Table 1. Manual and automatic scores of four MT systems taking part in the WMT10 evaluation campaign. Best results in bold. We report WMT manual evaluations (comparison with other systems and general understandability), our quiz-based evaluation and two automatic scores: BLEU and NIST. Note that the set of considered sentences was not identical across the evaluation methods for various reasons but the actual MT systems were.

user gets an overall picture of the output quality in the given test set and they can start by exploring the most frequent error types first.

The algorithms are described in more detail in (Zeman et al., 2011). The first step is the automatic detection of errors in the hypothesis translation. This is done by comparing tokens in the hypothesis with the reference translation, relying on some word alignment between the two texts.

Previous experience, i.a. (Bojar, 2011), shows that the quality of the alignment is critical, otherwise many errors can be mis-classified (e.g. a pair of "missing" and "extra" errors instead of one error of bad lexical choice). Our tool circumvents current limitations of word alignment by providing several methods and allowing the user to choose which of them works best for the particular language pair, MT system and dataset.

Currently, Addicter internally implements the alignments of WER (Levenshtein, 1966), LCS (Hunt and McIlroy, 1976), a greedy injective alignment, and an injective HMM (Fishel et al., 2011). The user can provide any additional alignment, e.g. GIZA++ (Och and Ney, 2003) alignments, or alignments obtained by linking a reference-to-source alignment with the source-to-hypothesis alignment as reported in a verbose output of the MT system (briefly called "via-source").

Addicter tries to automatically classify errors into categories similar to those of (Vilar et al., 2006), such as: morphological, reordering, missing words, extra words and lexical errors.

The errors can be then summarized into a table showing their counts in the test data. When using GUI, the table is connected to the test data browser, so with just one click, the user can see the list of sentences with the occurrence of the given error type and even look at the sentences one by one in more detail.

Test Set Origin	Number of Sentences	Number of References	Correlation BLEU vs. Manual Rank
WMT11 Official	3003	1	0.69
WMT11 Official	50	1	0.47
Post-edited MT Output	50	1	0.72
BLEU "Standard"	3003	4	0.74
Many Refs	50	500–50k	0.79

Table 2. Performance of BLEU (in terms of correlation with manual ranking of systems) depending on the test size (number of sentences) and the number of references per sentence.

2.5. Very Large Number of Reference Translations

It is well known that n-gram-based automatic evaluation methods notoriously depend on the number of available reference translations. A single reference, although most often used, cannot account for the range of correct possible outputs. For morphologically rich languages like Czech, the situation is worse; see also Bojar et al. (2010). The proper use of BLEU (Papineni et al., 2002) requires 4 references.

Dreyer and Marcu (2012) proposed to manually construct many, if not all, reference translations using a compact representation. In Bojar et al. (2013b), we aim at the same goal with Czech, which required us to completely redesign the framework to accommodate linguistic properties of Czech. The project CzechMATE then paid a few annotators to construct compact representations of many Czech references of just 50 English sentences. We confirm that it is easy to produce dozens or hundreds of thousands reference translations per one input sentence.

Table 2 summarizes our results. The correlation of manual ranking and BLEU with just one reference as provided for WMT evaluation campaigns is not very high: 0.69. If we restrict the testset to just the exact 50 sentences we use, the correlation drops to unacceptable 0.47. Having a reasonably sized test set is thus very important if only one reference is available; however, the situation seems to get much better if the 50 references are constructed by post-editing MT outputs and not translated independently: 0.72.

The proper use of BLEU (4 references over a 3000-set of sentences) performs acceptably: 0.74. We are nevertheless able to improve on this and reach 0.79 if we use many references. As we are adding references, the correlation steadily improves and it saturates around 500 reference translations selected randomly from the tens of thousands of possibilities. Due to the random selection, this 500-reference set is probably more diverse than if asked translators to produce 500 different translations.

We find the approach to MT evaluation which relies on (very) many reference translations and an automatic metric very promising since it mitigates most problems of manual MT evaluation (subjectivity, difficult replicability). Currently, con-

structing the many references is prohibitively expensive (2 hours per sentence) but we hope techniques to speed this up will emerge.

3. Improving Statistical MT into Czech

For a long time, published research on translation into Czech strongly focused on translation from English. Arguably, translation from different source languages may pose different problems. We picked a few other major European languages (German, Spanish and French) and compared English-Czech translation with translation from these languages. The choice was motivated practically both from the point of view of the user (expected likelihood that a user will need such translation) and of the developers (availability of training data). We are aware that the difference between English and any of these three languages is almost negligible when compared to the difference between English and any non-European language (e.g. Chinese). However, even our restricted experimental environment proves that individual, linguistically driven approach to every language pair is desirable and beneficial.

3.1. Core System

The findings presented in this article are based on experiments with various MT systems that differ from each other in their settings, combination of training data, pre- and postprocessing steps etc. Nevertheless, there is a core technology common to most of these systems. Unless explicitly stated otherwise, we use the Moses SMT decoder (Koehn et al., 2007) with baseline settings, Giza++ to compute word alignments (Och and Ney, 2003), the well-known *grow-diag-final-and* symmetrization heuristic (Koehn et al., 2003), the SRILM language modeling toolkit (Stolcke, 2002) and the minimum error-rate training algorithm, MERT (Och, 2003).

3.2. Data

We need two sorts of corpora for statistical machine translation: parallel data for translation models and monolingual data for target language models. The CzEng 1.0 parallel corpus (Bojar et al., 2012) provides a decent amount of English-Czech parallel data. It is less easy to obtain training data for German-to-Czech, Spanish-to-Czech and French-to-Czech translation.

We used the data provided for the annual WMT translation task (Callison-Burch et al., 2012)[1]: the Europarl corpus and the News Commentary parallel corpus. Both of the corpora contain text in each of the five languages we are interested in (Czech, English, German, Spanish and French). However, not all segments are available in all languages and the corpora are supplied as four Something-English pairs. Fortunately

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

there is a significant overlap across the pairs, so we were able to combine them. For instance, to create a Czech-German parallel corpus, we identified the intersection of the English sides of Czech-English and English-German corpora, respectively; then we combined the corresponding Czech and German sentences.

Table 3 shows the sizes of the corpora.

Corpus	SentPairs	Tokens lng1	Tokens lng2
en-cs	782,756	20,964,639	17,997,673
de-cs	652,193	17,422,620	15,383,601
es-cs	692,118	20,189,811	16,324,910
fr-cs	686,300	22,220,780	16,190,365
de-en	2,079,049	55,143,719	57,741,141
es-en	2,123,036	61,784,972	59,217,471
fr-en	2,144,820	69,568,241	59,939,548
CzEng en-cs	14,833,358	231,463,445	200,724,410

Table 3. Number of sentence pairs and tokens for every language pair in the Europarl, News Commentary and CzEng corpora. Every line corresponds to one language pair from combined Europarl and News Commentary, except for the line marked as CzEng. Languages are identified by their ISO 639 codes: cs = Czech, de = German, en = English, es = Spanish, fr = French.

Furthermore, there are test sets from six years of WMT shared tasks, about 3,000 sentences each. These are multi-parallel, available in all five languages. The contents are short news stories originally written in one of the languages (balanced) and human-translated to the others. We use these sets as development and test data; we hired translators from German to Czech to obtain four different Czech reference translations of the WMT 2011 set.[2]

The Czech side of the parallel corpora, especially CzEng, provides a decently sized monolingual corpus for training the target language model. Its size is still suboptimal for a morphologically rich and free-word-order language such as Czech. The situation would be much better for translation out of Czech, with Gigaword corpora (Parker et al., 2011) available for English, French and Spanish; unfortunately, there is no Czech Gigaword corpus yet. We use the Czech Crawled News corpus instead (also provided by WMT). It consists of 460 million tokens in 27.5 million segments (sentences).

All parallel and monolingual data underwent the same preprocessing. They were tokenized the same way, a few special characters were normalized or cleaned, and a set of language-dependent heuristics was applied in an attempt to restore and nor-

[2]These additional reference translations also served in our experiment described in Section 2.5 and they are freely available to other researchers, see `http://hdl.handle.net/11858/00-097C-0000-0008-D259-7`.

malize the opening/closing quotation marks (i.e. "quoted" → “quoted”). First, we hope that paired quotation marks could occasionally work as brackets and better denote parallel phrases for Moses; second, if Moses learns to produce directed quotation marks, subsequent detokenization will be easier.

Our heuristics applied to 1.84 % of Spanish sentences, 2.47 % Czech, 2.77 % German, 4.33 % English and 16.9 % French (measured on Europarl data). See Zeman (2012) for details.

We tag and lemmatize all texts. Lemmas are used to compute word alignment, and also to apply “supervised truecasing” (upper- or lowercase of the first letter of a word form is derived from its lemma). Without supervised truecasing, the models could not correctly utilize sentence-initial words, which are always uppercased. In Bojar et al. (2013c), we empirically confirm that this supervised truecasing indeed performs best for English-to-Czech translation. Morphological tags are used for delexicalized language modeling, to assess fluency of a morphologically rich language. We use the Featurama tagger for Czech and English lemmatization and TreeTagger for German, Spanish and French lemmatization. All these tools are embedded in the Treex analysis framework (Žabokrtský et al., 2008).

3.3. English as Pivot Language

Table 3 demonstrates that there are significantly more data for German / Spanish / French-English and for English-Czech translation, than directly for German / Spanish / French-Czech. Since more data typically means better models, one should ask whether we would not do better if we translated the source text first to English, then from English to Czech. There are also some obvious drawbacks of such an approach: since MT systems typically make errors, applying a system twice in a row could accumulate more errors. Every language has its own specific properties, be it fixed word order, morphological features or parts of speech that do not occur in other languages. Making the output consistent with these specifics is one of the biggest challenges that every MT system faces. Therefore, having first to accommodate constraints of an intermediate language could make the task unnecessarily difficult.

We ran several experiments to see which of the two approaches is better. Results are shown in Table 4. In terms of BLEU score, the results are not equally conclusive over all language pairs. The intermediate level clearly hurts German-Czech translation, to a lesser extent it also damages Spanish-Czech results. French-Czech seems to be (as far as BLEU can tell) the most difficult one among the investigated language pairs, and intermediate English does not change it (the fr-en-cs BLEU score is even higher than fr-cs but the difference is not statistically significant).

The table also shows human evaluation of the experiments, which provides a different picture. It suggests that both German and French are better translated via English (exploiting the big models). Spanish appears to be the least sensitive to the choice

src fr	les diabétiques ne seront plus tenus de contrôler leur taux de sucre .
fr-cs	na diabétiques nebudou povinny dohlížet na jejich cukru .
fr-en-cs	diabetiků nejsou povinny monitorovat jejich podíl cukru .
en-cs	diabetiků už nebudou muset kontrolovat hladinu cukru v krvi .
ref cs	diabetici si již nemusí hlídat hladinu cukru .
ref en	diabetics no longer need to control their blood sugar .

Figure 3. French to Czech direct or via English. The word diabétiques is OOV in the direct fr-cs model. The larger models with pivot English managed to find a Czech equivalent, even though they failed to pick the correct form (genitive diabetiků instead of nominative diabetici).

src de	Sie forderten weiterhin die Bildung von Gewerkschaften in der fast - Food - Branche .
de-cs	žádali i nadále vzdělání odbory v rychlé občerstvení - odvětví .
de-en-cs	žádali , aby mohli pokračovat v zakládání odborů ve fast - food industry .
en-cs	také požadovala vznik odborů ve fast food industry .
ref cs	také požadovali , aby ve sféře rychlého občerstvení byly založeny odborové organizace .
ref en	they also demanded the creation of unions in the fast food industry .

Figure 4. German to Czech direct or via English. Pivot English favored "zakládání odborů" (creation of unions) over "vzdělání odborů" (education of unions). Both creation and education are possible translations of German "Bildung". The direct model was more successful in translating "the fast food industry" but the overall fluency and understandability of the sentence is much better in the pivoted translation.

here; indeed, Spanish was the language where we did not identify many language-specific problems affecting translation into Czech.

Human inspection of the outputs revealed that English often helped to select better target words, or even cover source words that would be OOV (out-of-vocabulary) in the direct model. See the examples in Figure 3 and Figure 4.

3.4. Properties of Individual Source Languages

In this section we summarize linguistic characteristics that are specific to each investigated language pair and that influence the quality of MT output. For each specific phenomenon, we describe a linguistic transformation that, if applied both to the source-language part of the training data and of the test input, makes learning of the translation model easier.

	BLEU		Humans Prefer		
Pair	Direct	Via en	Direct	Neither	Via en
en-cs	0.1786				
de-cs	**0.1532**	0.1334	21	50	**29**
es-cs	**0.1614**	0.1570	**25**	57	18
fr-cs	0.1441	**0.1466**	25	39	**36**

Table 4. BLEU-score and manual evaluation of translation from various source languages to Czech. The figures in the first column evaluate direct models between the source language and Czech, trained on small data. The second column is via English, where much larger data is available for both steps (source to English and English to Czech). The last column shows the percentage of cases where human judgement scored the direct translation better, equally good (bad), or worse than the translation via English.

Czech is the target language in all four translation pairs; Czech has rich morphology (both inflectional and derivational) and relatively free word order. In fact, the predicate-argument structure, often encoded by fixed word order in English, is usually captured by inflection (especially the system of 7 grammatical cases) in Czech. The non-English source languages have freer word order and more morphology than English but still their morphology is much simpler than in Czech. Generating correct inflectional affixes is thus one of the main challenges of translation from any of these languages into Czech. Furthermore, the multitude of possible Czech word forms (significantly higher than in English) makes the data sparseness problem really severe.

The grammar of Czech requires two main layers of grammatical agreement: subject agrees with verb in person, number and gender, and adjective agrees with its governing noun in gender, number and case. (Furthermore, the choice of case is controlled by valency of verbs and prepositions.) Language models are typically too weak to enforce the agreement reliably. One of the most common translation errors is a wrong morphological form of otherwise correctly picked lemma.

A less pronounced difference between Czech and all the source languages is that Czech normally does not mark definiteness: there are no definite or indefinite articles. It is easy for the MT system to drop the articles; however, learning phrases with two different articles (or without any article) unnecessarily disintegrates statistics and makes the phrase table larger. Experiments show that dropping articles during preprocessing of the training data simplifies the models without decreasing the BLEU score. (Unlike in English, the German, Spanish and French articles also distinguish gender. It is not of much use for translation though, as the grammatical gender in the target language may differ.)

The following subsections present ideas how to adapt models to the individual source languages; these ideas have yet to be verified in experiments.

3.4.1. English to Czech

Czech is a pro-drop language, i.e. it is not required to supply a personal pronoun whenever there is no better subject in the sentence. However, Czech finite verbs are marked for person, which is much less visible in English. We can design a preprocessing step that will make sure that there is always a personal pronoun next to the English verb—even if there is a noun phrase functioning as subject. It will help the translation model to learn the correct Czech verb forms.

One area where the English language system is much more complex than the Czech one, is tenses. Czech has only three tenses (past, present and future). No perfect tenses (there are special perfective and imperfective verbs) and no progressive tenses. The periphrastic verb forms in English are a common source of translation errors. For instance, the auxiliary "is" (as in "he is doing") is sometimes translated to Czech, although it should not be used there. We thus need a preprocessing step that identifies the tense of the English verb and, if necessary, maps it to simple past, present or future. This way auxiliaries will be seen only with future and the trainer will find it easier to learn translation of content verbs.

In our combined system (Section 3.7), the complex English tenses are specifically handled by the deep-syntactic system TectoMT.

3.4.2. Spanish to Czech, French to Czech

Out of our language pool, Spanish and French possess the least grammatical peculiarities (but see the notes on negation in Section 3.4.4). Their word order is mostly compatible with preferred Czech word order, with one important exception: Adjectival modifiers usually follow the noun, in Czech they precede it.

3.4.3. German to Czech

German is genetically related to English (both belong to the Germanic group) and it has long history of close neighborhood and influencing of Czech. Nevertheless, it is distinctively different from both.

The uppercase / lowercase distinction is more important in German than in other languages because all German nouns (not just named entities) are capitalized.

Long German compound words are notorious for increasing out-of-vocabulary rate, which has led many researchers to devising unsupervised compound-splitting techniques. For instance, the word "Geschichtenerzähler" (storyteller) is OOV in our data; if we split it to "Geschichten" (stories) and "Erzähler" (teller), neither part will be OOV (see Section 3.5 for our approach to compound splitting).

German word order is not as fixed as English but there are strict rules about placement of verbs. Dependent verbal forms including participles are placed at the end of the clause, and the resulting long-distance dependencies often have deadly effect on meaning preservation, as demonstrated in Figure 5.

src de	französische Truppen haben ihren Verantwortungsbereich verlassen
lit. en	French troops have their responsibility-area left
ref en	French troops have left their area of responsibility
ref cs	francouzské jednotky opustily svou oblast odpovědnosti
de-cs	francouzské jednotky mají své povinnosti
lit.	French troops have their responsibility

Figure 5. Source German sentence uses the present perfect tense, "haben verlassen" (have left); however, the participle is placed far away from the finite form of "haben". The finite verb serves as auxiliary in this sentence but it could act as content verb elsewhere. Our model overlooked the participle at the end and took "haben" for content-bearing verb. As a result, "to leave responsibility area" was misinterpreted as "to have responsibility".

src es	en estos nuevos tiempos no es cómodo trabajar
lit. en	in these new times not is comfortable to-work
ref en	in the new times, it is uncomfortable to work
es-cs	v této nové době je příjemné pracovat
lit. en	in this new time is pleasant to-work
ref cs	v nových časech je práce nepříjemná

Figure 6. Negation lost in translation. Czech negation is typically marked by a morpheme bound to the verb, not by a separate particle. In this case, the words "no es" should be translated as one word "není", not "je" (underlined); however, since there are various other means to express the negative meaning (one of them instantiated in the Czech reference here), the model learned that the negative particle "no" often remains unaligned. It is a dangerous observation and leads to dropping of negation quite frequently.

The problem can be solved by moving participles back to the auxiliaries during preprocessing of the data, see i.a. Collins et al. (2005). Similarly, one may also want to move separable verb prefixes closer to the corresponding verb stems.

3.4.4. Negation

The various linguistic devices for expressing negation pose a separate set of problems. It is possible to generate a perfectly fluent sentence with 95 % words translated correctly, yet with the overall meaning totally reversed (e.g. Figure 6) – this is also one of the reasons of low reliability of many automatic MT evaluation methods.

Czech negation is typically marked using the prefix "ne-" on verbs or adjectives (example: "Student **ne**přišel.") In English, the auxiliary verb "to do" is usually needed

	BLEU		Humans Prefer		
Pair	Words	Morphs	Words	Neither	Morphs
en-cs	**0.1632**	0.1425	29	**42**	29
de-cs	**0.1532**	0.1272	24	33	**43**
es-cs	**0.1614**	0.1344	**31**	45	24
fr-cs	**0.1441**	0.1186	**31**	40	29

Table 5. BLEU-score and manual evaluation of translation using a word-based model (default) and a morpheme-based model. All models were trained on combined News Commentary and Europarl (no CzEng), additional language model was trained on the Czech Crawled News corpus. Morphemes were recombined to words before evaluation of the morpheme-based models. The three columns to the right show the percentage of cases where human judgment scored the word-based translation better, equally good (bad), or worse than the morpheme-based translation.

("The student **did not** come.") But adjectives behave the same way as in Czech: the prefix **un-** is a bound morpheme. In German and Spanish there is a negative particle but no auxiliary verb is needed ("Der Student kam **nicht.**" "El estudiante **no** llegó.") French is different in that it has two negative particles ("L'étudiant **n'**est **pas** venu.")

As with other peculiarities, the training situation can be improved using a cheap trick: separate the Czech negative prefixes so that they are learned as separate words. It will reduce the number of occurrences of unaligned negative particles on the source side. In English, the auxiliary "does / do / did" could be removed (see also progressive tenses in Section 3.4.1).

There is still one problem open, though: it is not rare in Czech to see negation marked on several words in the same clause ("**nikdy ne**přišel **žádný** student" = lit. "never not-came no student"). Such a Czech sentence is difficult to generate because in our source languages the negation is typically marked only once ("**no** student ever came", "the student **never** came", "**ningún** estudiante llegó" etc.) Sentences such as "*__Žádný__ student přišel." are not grammatically correct Czech. One could attempt to recognize all "negatable" words in the preprocessing phase and negate their source-language version. It is difficult though to identify the exact set of such words and to produce their negated form reliably.

3.5. Morphemic Segmentation of Words

There are many long compounds in German. Many are OOV (unknown from training data, appear in test data) and the set of possible compounds is in theory infinite. It is therefore desirable to split compounds to individual stems during the preprocessing phase. We decided to go a step further and to approach morphological forms in all the languages the same way. We first segment all training words into morphemes,

src es	aquí vislumbramos las premisas de una teocracia » .
ref en	this suggests the beginnings of a theocracy . ”
es-w-cs	zde vislumbramos předpoklady k teokracii . “
es-m-cs	zde vidíme výchozí teokracii “ .
ref cs	lze v tom spatřovat začátky teokracie “ .

Figure 7. Spanish “vislumbramos” (“we see/sense”) is unknown to the word-based model but the morpheme-based model succeeds in decomposing and translating it. It has been segmented as “vislumbr/STM +a/SUF +mos/SUF”. The phrase table contains 700 entries with the stem “vislumbr” but none of them is the 1st person plural “vislumbramos”.

src de	in der römischen Zeit war Caesarea die wichtigste Stadt Judäas
ref en	during the Roman period , Caesarea was the main city of Judea
de-w-cs	v římské době bylo Caesarea hlavní město Judäas
de-m-cs	v římské době bylo Caesarea hlavní město Judäasské
ref cs	v římských dobách byla Caesarea hlavním městem Judeje

Figure 8. The morpheme-based model constructed adjective from Judea. This experiment did not use the model for named entities described later in this article, so Judäa remained in its German spelling but the Czech adjectival suffix “ské” was attached to it.

then learn a translation model from the parallel sequences of morphemes. We hope to model morphological behavior of the other languages too. For example, locative expressions will often be translated to Czech using the locative case. While the system is likely to observe all possible locative suffixes in the training data, it is much less likely to encounter all words in all cases (including locative). But it may be able to combine a known stem with a known locative suffix and create a valid word form, which has not itself occurred in the training data.

We use the freely available tool Morfessor (Creutz and Lagus, 2007) to segment all corpora into morphemes. In addition to segmentation, Morfessor also classifies the morphemes as prefixes, stems and suffixes, respectively. To give an example, the German phrase “aus dem Strafgesetzbuch zu entfernen” (“to remove from the Criminal Code”) is broken down into “aus/STM dem/STM straf/PRE+ gesetz/STM + buch/STM zu/STM ent/PRE+ fernen/STM”. The phrases that Moses learns are sequences of tagged morphemes, thus the Moses decoder also generates similar sequences. We take the generated morphemes, remove their tags and join them on plus signs to get full words again.

src de	das heißt , dass Ibrahimovic leistungsstark ist und viele Tore schießt .
ref en	this means Ibrahimovic is very successful and scores a lot of goals .
de-w-cs	to znamená , že Ibrahimovic výkonná je mnoho a střílí góly .
de-m-cs	to znamená , že ibrahim , ovic silný a mnoho gól střílí .
ref cs	to znamená , že Ibrahimovič je velmi výkonný a že dává hodně gólů .

Figure 9. Sometimes the generated morphemes do not allow for correct rejoining of the word.

src en	diabetics no longer need to control their blood sugar .
en-w-cs	diabetiky už třeba kontrolovat jejich krevního cukru .
en-m-cs	diabetiky už nepotřebují kontrolovat jejich krevního cukru .
ref cs	diabetici si již nemusí hlídat hladinu cukru .

Figure 10. Negation is preserved by the morpheme-based model while the word-based model destroys it.

Table 5 evaluates experiments with morpheme-based translation models. The BLEU scores are disappointing: segmentation to morphemes consistently and significantly hurts the results across all source languages. However, part of the decrease could probably be attributed to the known imperfections of the BLEU metric, as it does not always correlate with human judgment. The difference between words and morphemes is much less pronounced under human evaluation, and in German-to-Czech translation morphemes are even preferred over words. We have not tested the partial model where only the source-language text is segmented.

Figures 7 through 10 portray more details about the pros and cons of the two approaches.

3.6. Named Entities

There is no single and clear-cut definition of what consists a NAMED ENTITY (a name) in a text. It is nevertheless obvious that a high-quality MT system has to address names somehow specifically. Pure cooccurrence statistics in parallel data may be rather deceiving, see the following output of Google Translate:

(1a) Source: Doktor Novák také přišel.
(1b) Google: Dr. Smith also attended.
(1c) Reference: Dr. Novák also came.

Novák, being the most frequent Czech name, is often translated as *Smith* in literary texts. In most other types of text, names should be preserved.

(2a) Source: Sejdeme se v Plzni.
(2b) Google: Meet me in London.
(2c) Reference: Let's meet in Pilsen.

This second example is just obscure and reveals the level of noise in Google's parallel data. The actual output differs if we omit the full stop in the source sentence. (Outputs show as of January 2014.)

For our experiments, as detailed in Hálek et al. (2011), we defined a NAMED ENTITY as *a word or group of words which, when left untranslated, are a valid translation anyway*. Some named entities, esp. geographical names and names of institutions, have translations. Depending on the salience of the item, the translation can be preferred very strongly (*Paris–Paříž*), less so (*Trier–Trevír*) or it can become even confusing to people who do not know the specifics (*Görlitz–Zhořelec*), in which case the system should probably produce both variants of the name. For most named entities, there is no counterpart in the other language.

Even if the named entity is not translated, morphologically rich languages like Czech require the entity to be adapted for the context of the sentence (*We met in Trier–Sešli jsme se v Trieru*, or to be introduced with a common noun describing the entity type (*I bought this in IKEA.–Koupil jsem to v obchodním domě IKEA.*, lit. "in the shopping mall IKEA"). While the latter option is also very interesting from the technical point of view (automatically adding the descriptive noun phrase), we attempted to improve the translation and declension of named entities.

We used Wikipedia as the natural source for named entities. This is the outline of our procedure:

1. Search for named entities in the source text. We preferred our simple recognizer based on letter case due to its higher recall over an established recognizer of a higher precision.
2. Confirm entities in Wikipedia. The potential entities need to be present in English Wikipedia and the category of their article has to fall among those that we consider named entities, e.g. Places, People, Organizations.
3. Extract base translation from Wikipedia. We simply follow the cross-language link from the Wikipedia article to find the lexical form of the named entity.
4. Extract variants of the translation. We use the Czech page and collect all phrases with stems identical to the stems of the lexical variant of the named entity. Each of the variant gets a score based on its frequency in the page.
5. Extend the list of "translation options" for the source phrase with all the extracted variants of the translation.

The experiment revealed that automatic MT evaluation disprefers our changes but manual evaluation suggests that both translating entities using our procedure (preferred in 34% cases) as well as identifying them and preserving untranslated (37%) is preferred over the baseline (preferred in 29% of cases).

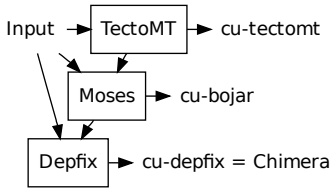

Figure 11. Chimera, a combination of three approaches to English-to-Czech MT.

3.7. Best of All Worlds: System Combination

It has already been established that "system combination" (Matusov et al., 2008), i.e. MT output constructed by combining outputs of several primary MT systems, outperforms the individual systems (Callison-Burch et al., 2009, 2010, 2011).

Our system Chimera described in Bojar et al. (2013c) is constructed differently from the standard system combination techniques. Instead of collecting complete outputs of several systems and selecting words that the majority of systems produced, we combine three different approaches to English-to-Czech MT in a sequential way as depicted in Figure 11.

Chimera consists of the following components:

TectoMT (Žabokrtský et al., 2008; Galuščáková et al., 2013) is a hybrid transfer-based MT system that processes English to obtain deep syntactic trees, transfers them to Czech trees using a statistical model and generates Czech sentence using a set of rules. Aside from the standard Czech morphological dictionary, TectoMT also includes a basic derivational dictionary so it can e.g. derive translations of adverbs even if only the corresponding adjectives were seen in parallel data. The target-side vocabulary of TectoMT is thus not restricted to observed items. Further gains can be expected from linguistically adequate handling of complex verbs.

The output of TectoMT *for the given input sentences*, i.e. for the source side of the test set, is included in the training data of the following step.

Moses (Koehn et al., 2007) is a standard phrase-based and hierarchical MT toolkit. It serves as the central component of Chimera, producing its single-best hypothesis. Trained on the large corpus CzEng (Bojar et al., 2012), data prepared by the organizers of WMT13[3] and about 3.6 gigaword of Czech news in the language model, our configuration is a strong system on its own.

The additional synthetic training data as provided by TectoMT allow Moses to produce words never seen in the parallel training data. The weights selecting whether to prefer translation of phrases from the parallel data or the output of

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

System	Automatic		Manual WMT Ranking		
	BLEU	TER	Appraise	MTurk	Total
CU-TECTOMT	14.7	0.741	0.455	0.491	0.476
CU-BOJAR	**20.1**	0.696	0.637	**0.555**	**0.580**
CU-DEPFIX = Chimera	20.0	**0.693**	**0.664**	0.542	0.578
Plain Moses	19.5	0.713	–	–	–
Google Translate	19.4	0.720	0.618	0.526	0.562

Table 6. Results of Chimera, an English-to-Czech system that combines TectoMT, Moses and Depfix in a sequential way. Best scores in bold.

TectoMT are automatically optimized on a heldout dataset. More details on the tuning are available in Bojar et al. (2013c).

Depfix (Rosa et al., 2012) is a system for correcting errors in (esp. phrase-based) MT outputs. As all components of Chimera, it makes use of the original English sentence to reduce the problem of error cummulation. Depfix parses both Moses output and the original input, fixing the former on the basis of the latter if needed. Hand-written rules then specify which words in the dependency tree need to agree in grammatical categories such as case or gender with the target language, or match with the source in number or negation.

Table 6 presents the results of Chimera on WMT13 test set using two automatic measures (BLEU and TER) and the official manual ranking. The manual ranking was performed by two groups of people: researchers and their colleagues (labelled "Appraise" after the annotation frontend) and random paid annotators using Amazon Mechanical Turk crowdsourcing platform ("MTurk" in the table). Note that the actual annotation interface was identical for both of the groups. The official results of WMT13 (Bojar et al., 2013a) do not distinguish between the annotator groups and we list the scores in the column "Total".

We submitted all the three steps of Chimera as independent systems to the evaluation. A big jump in quality comes with the powerful statistical Moses. Measured by both automatic metrics, TectoMT is an important component of the mix, raising BLEU of 19.5 ("Plain Moses", i.e. the same setup except without access to the output of TectoMT) to 20.0 (CU-BOJAR).

The impact of Depfix seems less clear. Since Depfix alters only a few words in each sentence, it is not surprising that the automatic scores do not change much. Depfix also does not use any language models, losing a little in the n-gram-based evaluation by BLEU.

The most interesting is the discrepancy in manual scores of Turkers and researchers. Turkers prefer the outputs without Depfix while researchers clearly appreciate corrected outputs better (0.637 vs. 0.664). We speculate that Turkers, rewarded for each submitted item of annotation, hurried up and evaluated the outputs more superfi-

cially. The critical little differences corrected by Depfix (most notably lost negation) may have often remained unnoticed. As Bojar et al. (2013a) reports in Table 3, the inter-annotator agreement among Turkers evaluating Czech was exceptionally low this year, reaching only κ of 0.075 compared to 0.408 for researchers or to 0.25 which is the average κ of Turkers across all language pairs.

Given the low reliability of Turker judgements, we conclude that Depfix does play an important role in the final translation quality.

Overall, Chimera outperfomed Google Translate in both automatic and manual scores, ranking first among the English-to-Czech systems.

3.8. Fully Automated Research in MT Not Feasible

As apparent from the previous sections, MT systems are very complex cascades of processing steps, each of which is influenced by various parameters. Finding the right configuration of these parameters is critical for system performance. Some of the parameters have the form of a real value or a vector of real values. For these, we follow the common practice and use a variant of a grid search. However, many settings are categorical or stand outside of the standard automatic search: for instance, there are several different algorithms available for the search itself.

We see research as the search for the best (design and) configuration of a complex system. We developed tools that support this vision and allow for manual or even fully automatic search in the space of MT system configuration.

The core of our tools is EMAN (Bojar and Tamchyna, 2013), a generic experiment manager that promotes to represent experiments as acyclic graphs of basic processing steps. EMAN facilitates the creation of the individual steps and most importantly the derivation of steps and whole experiments by altering existing ones. For MT, our EMAN steps correspond to tasks such as word alignment, extraction of translation or language models, model optimization or translation of unseen texts. EMAN includes an assistant for organizing the obtained results and makes it easy to handle dozens or hundreds of experiments without losing focus.

EMAN also served as the basis of *fully automatic* search for the best MT system. With EMAN, it is easy to examine the full Cartesian product of various settings or to navigate in this space using grid search or e.g. genetic algorithms. In Tamchyna and Bojar (2013), we however document that the domain of machine translation is too complex to be examined automatically, even when restricted to the single model of phrase-based translation and one particular language pair. There are two reasons of this infeasibility:

Too large space of configurations. In our experiments, we included source and target-side features from the morphological, surface and deep syntactic analysis of the sentence. Picking which features to choose and how to use them allows for more than a thousand possible configurations even in a very restricted experiment de-

sign. Each of these configurations would require new extraction of translation tables and a new model optimization, which is quite computationally expensive.

Too imprecise evaluation. The main problem, however, is the impossibility to tell better and worse systems apart, especially if the difference between them is not very big. The optimization of a given configuration is a randomized process leading to various results. Multiple runs of the examined configurations led to scores so different that it was possible to completely reverse the ranking of some of the configurations. In short, the resulting scores are similar and their variance is high.

To conclude, it is not possible to find the best system configuration fully automatically, but EMAN can at least support researchers in their semi-automatic examination of the space.

4. Conclusion

In the present article, we summarized the results of the project CzechMATE, which focused on statistical machine translation into Czech. Four different source languages were examined: English, German, Spanish and French.

Throughout the project, we contributed to the state of the art in several ways, ranging from techniques of manual and automatic MT evaluation over comparison of direct translation and pivoting through English (e.g. translation from German to Czech via English, where pairwise parallel data are easier to obtain), translation of words broken into unsupervised morphemes to experiments with handling named entities. We conclude by describing our combined MT system called Chimera that obtained very high ranking in the WMT 2013 shared task and by attempts to (fully) automate research in MT.

Acknowledgements

The work on this project was supported by the grant P406/11/1499 of the Czech Science Foundation (GAČR).

Bibliography

Berka, Jan, Martin Černý, and Ondřej Bojar. Quiz-Based Evaluation of Machine Translation. *Prague Bulletin of Mathematical Linguistics*, 95:77–86, March 2011. ISSN 0032-6585.

Berka, Jan, Ondřej Bojar, Mark Fishel, Maja Popović, and Daniel Zeman. Tools for machine translation quality inspection. Technical Report TR-2013-50, Univerzita Karlova v Praze, MFF, ÚFAL, Praha, Czechia, 2013. URL `http://ufal.mff.cuni.cz/techrep/tr50.pdf`.

Birch, Alexandra, Barry Haddow, Ulrich Germann, Maria Nadejde, Christian Buck, and Philipp Koehn. The Feasibility of HMEANT as a Human MT Evaluation Metric. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 52–61, Sofia, Bulgaria,

August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2203.pdf.

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

Bojar, Ondřej and Aleš Tamchyna. The Design of Eman, an Experiment Manager. *The Prague Bulletin of Mathematical Linguistics*, 99:39–58, 2013. ISSN 0032-6585.

Bojar, Ondřej and Dekai Wu. Towards a Predicate-Argument Evaluation for MT. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 30–38, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-4204.

Bojar, Ondřej, Kamil Kos, and David Mareček. Tackling Sparse Data Issue in Machine Translation Evaluation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 86–91, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-2016.

Bojar, Ondřej, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

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 *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2201.

Bojar, Ondřej, Matouš Macháček, Aleš Tamchyna, and Daniel Zeman. Scratching the Surface of Possible Translations. *LNCS*, 2013b. ISSN 0302-9743.

Bojar, Ondřej, Rudolf Rosa, and Aleš Tamchyna. Chimera – Three Heads for English-to-Czech Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 92–98, Sofia, Bulgaria, August 2013c. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2208.

Callison-Burch, Chris, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W07/W07-0218.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, Athens, Greece, March 2009. 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.aclweb.org/anthology/W10-1703. Revised August 2010.

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. URL http://www.aclweb.org/anthology/W11-2103.

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. URL http://www.aclweb.org/anthology/W12-3102.

Collins, Michael, Philipp Koehn, and Ivona Kučerová. Clause restructuring for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540, Ann Arbor, Michigan, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219906.

Creutz, Mathias and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TLSP)*, 4(1)(3), 2007. URL http://dl.acm.org/citation.cfm?id=1187418.

Dreyer, Markus and Daniel Marcu. HyTER: Meaning-Equivalent Semantics for Translation Evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1017.

Fishel, Mark, Ondřej Bojar, Daniel Zeman, and Jan Berka. Automatic Translation Error Analysis. In *Text, Speech and Dialogue: 14th International Conference, TSD 2011*, volume LNAI 3658. Springer Verlag, September 2011. ISBN 3-540-28789-2.

Galuščáková, Petra, Martin Popel, and Ondřej Bojar. PhraseFix: Statistical post-editing of TectoMT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 141–147, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2216.

Hálek, Ondřej, Rudolf Rosa, Aleš Tamchyna, and Ondřej Bojar. Named Entities from Wikipedia for Machine Translation. In Lopatková, Markéta, editor, *ITAT 2011 Information Technologies – Applications and Theory*, volume 788, pages 23–30, September 2011. ISBN 978-80-89557-02-8.

Hunt, James W. and M. Douglas McIlroy. An Algorithm for Differential File Comparison. Computing Science Technical Report 41, Bell Laboratories, June 1976.

Koehn, Philipp. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In Frederking, Robert E. and Kathryn Taylor, editors, *AMTA*, volume 3265 of *Lecture Notes in Computer Science*, pages 115–124. Springer, 2004. ISBN 3-540-23300-8.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association*

for Computational Linguistics on Human Language Technology, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462.

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 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Praha, Czechia, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-2045`.

Levenshtein, Vladimir Iosifovich. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

Lo, Chi-kiu and Dekai Wu. Meant: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 220–229, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1023`.

Lopatková, Markéta, Zdeněk Žabokrtský, and Václava Kettnerová. *Valenční slovník českých sloves*. Univerzita Karlova v Praze, Nakladatelství Karolinum, Praha, 2008. In cooperation with Karolína Skwarska, Eduard Bejček, Klára Hrstková, Michaela Nová and Miroslav Tichý.

Matusov, Evgeny, Gregor Leusch, Rafael E. Banchs, Nicola Bertoldi, Daniel Dechelotte, Marcello Federico, Muntsin Kolss, Young-Suk Lee, Jose B. Marino, Matthias Paulik, Salim Roukos, Holger Schwenk, and Hermann Ney. System Combination for Machine Translation of Spoken and Written Language. *IEEE Transactions on Audio, Speech and Language Processing*, 16(7):1222–1237, September 2008.

Och, Franz Josef. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075096.1075117.

Och, Franz Josef and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Panevová, Jarmila. Valency Frames and the Meaning of the Sentence. In Luelsdorff, Ph. L., editor, *The Prague School of Structural and Functional Linguistics*, pages 223–243, Amsterdam-Philadelphia, 1994. John Benjamins.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2002. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073083.1073135.

Parker, Robert, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English Gigaword fifth edition, LDC2011T07, June 2011. URL `http://catalog.ldc.upenn.edu/LDC2011T07`.

Popel, Martin and Zdeněk Žabokrtský. TectoMT: Modular NLP Framework. In Loftsson, Hrafn, Eirikur Rögnvaldsson, and Sigrun Helgadottir, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 7th International Conference on Advances in Natural Language Processing (IceTAL 2010)*, volume 6233 of *Lecture Notes in Computer Science*, pages 293–304, Berlin / Heidelberg, 2010. Iceland Centre for Language Technology (ICLT), Springer. ISBN 978-3-642-14769-2.

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, June 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W12-3146`.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands, 1986.

Stolcke, Andreas. SRILM – an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado, USA, 2002.

Tamchyna, Aleš and Ondřej Bojar. No Free Lunch in Factored Phrase-Based Machine Translation. In *Proc. of CICLing 2013*, volume 7817 of *LNCS*, pages 210–223, Samos, Greece, 2013. Springer-Verlag.

Vilar, David, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. Error Analysis of Machine Translation Output. In *International Conference on Language Resources and Evaluation*, pages 697–702, Genova, Italy, May 2006.

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. ISBN 978-1-932432-09-1.

Zeman, Daniel. Data issues of the multilingual translation matrix. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 395–400, Montréal, Canada, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6.

Zeman, Daniel, Mark Fishel, Jan Berka, and Ondřej Bojar. Addicter: What is wrong with my translations? *The Prague Bulletin of Mathematical Linguistics*, 96:79–88, 2011.

Address for correspondence:
Daniel Zeman
`zeman@ufal.mff.cuni.cz`
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
CZ-11800 Praha, Czechia

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 97–110

A computational study on preverbal and postverbal accusative object nouns and pronouns in Ancient Greek

Giuseppe G. A. Celano

Tufts University, USA

Abstract

Many studies try to determine whether Ancient Greek is an OV or VO language. All of them, however, fail to conduct a research whose method is entirely clear. This paper presents the first attempt to quantify the number of verbs governing preverbal or postverbal accusative object nouns or pronouns in single or coordinate independent clauses in Homer's Iliad and Odyssey, Herodotus' Histories, and the New Testament, by providing results which are fully verifiable and reproducible. I prove that as for the parameter OV vs. VO there is great variation in the texts, which suggests a change over time from OV order in Homer to VO order in the New Testament. The figures for Herodotus' Greek prove a quasi-exact match between OV order and VO order.

1. Introduction

Ancient Greek (AG) is an Indo-European language allowing great freedom of word order at both clausal and subclausal level. A great variety of studies were conducted on the position of subject (S), verb (V), and object (O) to establish the "normal" order of such constituents (see, among others, Ebeling (1902); Friederich (1975); Cervin (1990); Kwong (2005)). They however provide discordant results, which are impossible to evaluate (see, for example, Cervin (1990); Taylor (1994)): the sample analyzed is often limited and, what is worse, the method employed to count the instances of a given word order is usually not precisely defined: e.g., Friederich (1975) counted 195 constructions in Iliad 5.1–296, but it is not clear what exactly he means by a construction.

© 2014 PBML. All rights reserved. Corresponding author: giuseppe.celano@tufts.edu
Cite as: Giuseppe G. A. Celano. A computational study on preverbal and postverbal accusative object nouns and pronouns in Ancient Greek. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 97–110.
doi: 10.2478/pralin-2014-0006.

Even when the criteria for counting seem to be clearer, the amount of data to manually process is so large that the research is unlikely to be reproducible (see, for example, Kwong (2005)). The present study, therefore, aims to be the first attempt at scientific quantification of AG word order: more precisely, I search the Ancient Greek Dependency Treebank (AGDT)[1] and the Pragmatic-Resources-in-Old-Indo-European-Languages corpus (henceforth the PROIEL Treebank, PROIELT)[2] for verbs governing preverbal or postverbal accusative object nouns or pronouns in single or coordinate independent clauses in Homer's Iliad and Odyssey, and the annotated parts of Herodotus' Histories and the New Testament (henceforth also referred to as OV and VO order, respectively).

In Section 2, I describe the data on which the research is based, i.e., how the morphosyntactic annotation of the texts has been encoded in the AGDT (Section 2.1) and the PROIELT (Section 2.2). Section 3 shows the method of the research: in Section 3.1 and 3.2 the queries used to search the AGDT and the PROIELT, respectively, are presented and explained. The results of the queries are given in Section 4 and discussed in Section 5. Section 6 summarizes the article.

2. Data

The data of the present research are drawn from the AGDT and the PROIELT, the two open source dependency treebanks currently available for AG. The AGDT provides Homer's Iliad and Odyssey, while Herodotus' Histories and the New Testament are contained in the PROIELT. The three texts belong to different varieties and stages of the language: the Homeric poems are likely to have been created in the 8th century BC; the historian Herodotus lived approximately between 485 and 424 BC; the New Testament was written in the 1st century AD.

All the texts have been automatically divided into sentences on the basis of the punctuation of the source text. Each word of such sentences has been semi-automatically annotated for morphology and manually annotated for syntax. The categories for the morphological annotation are those elaborated by traditional grammar (see more in Section 2.1 and 2.2): e.g., the noun is annotated for number, gender, and case, while the verb is annotated for number, person, tense, mood, and voice.

The syntactic annotation, for which specific guidelines (Bamman and Crane (2008); Haug (2010)) exist, mostly relies on the Prague Dependency Treebank 2.0 (PDT 2.0) Manual for Analytic Annotation (Hajičová et al. (1999)). Although the annotation style of the AG treebanks is different (see Section 2.1 and 2.2), they share the same core structure: according to the theory of Dependency Grammar as developed for the analytical level of the PDT 2.0, each text is a set of sentences, and each sentence is a set of words connected to each other in a head-dependent relationship.

[1] http://nlp.perseus.tufts.edu/syntax/treebank/greek.html

[2] http://foni.uio.no:3000/users/sign_in

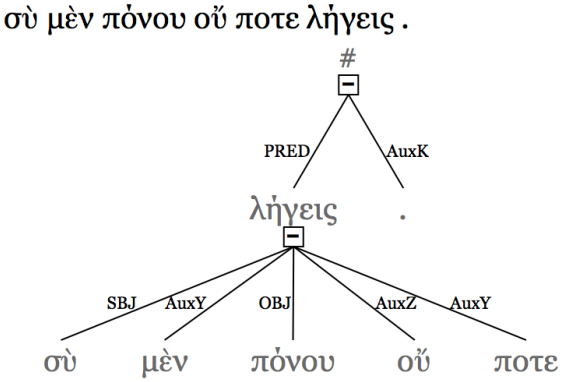

Figure 1. Example of a dependency tree from the AGDT

Each sentence can be represented as a top-down tree where the topmost word (typically the main verb or a coordinate conjunction) directly or indirectly governs the whole structure (excluding the final punctuation mark):

Example 1

σὺ	μὲν	πόνου	οὔ	ποτε	λήγεις
you.NOM.SG	PTC	work.GEN.SG	not	ever	cease.PRS.IND.ACT.2SG

'You never cease from working'

(Il. 10.164)

As Figure 1 shows, each word of Example 1 corresponds to a node, which is associated with its grammatical function: e.g., πόνου is annotated as an object (OBJ) of the predicate λήγεις, which takes the function PRED, i.e., "predicate", because it is the (main) verb of the sentence. Note that a word can have only one head, but a head can have more than one dependent: e.g., the verb λήγεις has five dependents.

The function OBJ, as well as the function SBJ, marks the arguments of a verb (for a more detailed definition see Sgall et al. (1986); Hajičová et al. (1999)). Depending on the meaning of the verb, the function OBJ can be conveyed not only by different cases in AG, but also by different parts of speech (e.g., nouns and adverbs) and constructions (e.g., infinitive and participial clauses). For the purposes of the present research, I restrict the analysis to only OBJ constituents being accusative nouns or pronouns in single or coordinate independent clauses.

It is important to underline that from the perspective of linguistic theory two different kinds of dependency are captured in the syntactic annotation: grammatical dependencies and "technical" dependencies. The former correspond to (rather) uncon-

troversial linguistic dependencies, such as that between the verb and its subject, while the latter concern dependencies such as coordination, whose annotation is rather governed by theory-dependent rules. In both the AGDT and the PROIELT, a coordinate conjunction is taken to technically govern its conjuncts and depend on the word to which such conjuncts are grammatically related. In the following two sections, I will present the AGDT and the PROIELT: the reader is referred to the online documentation and XML files for a full description.[3]

2.1. The AGD Treebank

The AGDT includes Homer's Iliad and Odyssey (232.339 words, including punctuation). As an illustration, the following text shows how Example 1 is coded in the XML file:

```
<sentence subdoc="urn:cts:greekLit:tlg0012.tlg001:10.164" id="2277214"
          document_id="Perseus:text:1999.01.0133" span="σὺ0:.7">
  <primary>alexlessie</primary>
  <primary>millermo</primary>
  <secondary>nicanor</secondary>
  <word id="1" cid="34282848" form="σὺ" lemma="σύ1" postag="p-s----n-" head="6"
        relation="SBJ" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="2" cid="34282849" form="μὲν" lemma="μέν1" postag="g--------" head="6"
        relation="AuxY" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="3" cid="34282850" form="πόνου" lemma="πόνος1" postag="n-s---mg-" head="6"
        relation="OBJ" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="4" cid="34282851" form="οὔ" lemma="οὐ1" postag="d--------" head="6"
        relation="AuxZ" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="5" cid="34282852" form="ποτε" lemma="ποτέ1" postag="g--------" head="6"
        relation="AuxY" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="6" cid="34282853" form="λήγεις" lemma="λήγω1" postag="v2spia---" head="0"
        relation="PRED" cite="urn:cts:greekLit:tlg0012.tlg001:10.164"/>
  <word id="7" cid="34282854" form="." lemma="περιοδ1" postag="u--------" head="0"
        relation="AuxK" cite=""/>
</sentence>
```

Each <sentence> element has some attributes identifying it: particularly relevant are the @id attribute, which specifies the number of the sentence, and the @subdoc attribute, which specifies the reference of the sentence. Within the <sentence> element there are two kinds of children. The <primary> and <secondary> elements contain the names of the annotators of the sentence: the primary ones annotated the sentence independently of each other, with the secondary one deciding when the primary annotations were different. The second kind of children are the <word> elements: there are as many as the actual words of the sentence. Each <word> element has 8 attributes:

- The value of the @id attribute indicates the position of the word in the sentence.
- The value of the @cid attribute is the unique number of the word in the database.

[3] The relevant documentation and the exact files on which I performed my queries can be downloaded at https://github.com/gcelano/PBML101_2014.

- The value of the @form attribute is the actual form of the word in the sentence.
- The value of the @lemma attribute corresponds to the dictionary form of the word.
- The value of the @postag attribute is the morphological annotation of the word: it consists of a string of 9 ordered positions, each of which can be filled by a character or, if a value is missing, a hyphen. The 9 positions correspond to: part of speech (1), person (2), number (3), tense (4), mood (5), voice (6), gender (7), case (8), and degree (9). For example, the postag value n-s---mg- of the word πόνου means noun (n), singular (s), masculine (m), genitive (g).[4]
- The value of the @head attribute coincides with the value of the @id attribute of the head word: e.g., the word πόνου is syntactically annotated as depending on the word λήγεις because the value of its @head attribute and the value of the @id attribute of λήγεις match (6 in both). If the head value of a word is 0, this means that the word has no head.
- The value of the @relation attribute indicates the kind of relation the word entertains with its head: e.g., the word πόνου has an OBJ relation, which means that it is taken to be an object with respect to the predicate λήγεις, its head (see Bamman and Crane (2008) for the list and meaning of all relations).
- The value of the @cite attribute shows the exact reference of the word.

2.2. The PROIEL treebank

The PROIEL corpus is a parallel corpus containing the morphosyntactic annotation of almost all the New Testament in AG (124.991 tokens) and its translations in Old Church Slavonic, Classical Armenian, Gothic, and Latin. The corpus also includes the annotation of some literary works, such as (part of) Herodotus' Histories (71.719 tokens). They are made available in XML format. The following is an example of how a sentence of the New Testament is encoded in the PROIELT:

```
<sentence id="47594" status="reviewed" presentation-after=" ">
 <token id="266750" form="Ἀμιναδὰβ" citation-part="MATT 1.4" lemma="Ἀμιναδάβ"
        part-of-speech="Ne" morphology="---------n" head-id="266752" relation="sub"
        antecedent-id="266748" information-status="old" presentation-after=" "/>
 <token id="266751" form="δὲ" citation-part="MATT 1.4" lemma="δέ"
        part-of-speech="Df" morphology="---------n" head-id="266752" relation="aux"
        presentation-after=" "/>
 <token id="266752" form="ἐγέννησεν" citation-part="MATT 1.4" lemma="γεννάω"
        part-of-speech="V-" morphology="3saia----i" relation="pred"
        presentation-after=" "/>
 <token id="266753" form="τὸν" citation-part="MATT 1.4" lemma="ὁ"
        part-of-speech="S-" morphology="-s---ma--i" head-id="266754" relation="aux"
        presentation-after=" "/>
 <token id="266754" form="Ναασσών" citation-part="MATT 1.4" lemma="Ναασσών"
        part-of-speech="Ne" morphology="---------n" head-id="266752" relation="obj"
```

[4]The full value list can be found at https://github.com/gcelano/PBML101_2014.

```
        information-status="acc_gen" presentation-after=","/>
</sentence>
```

The text has been divided into sentences corresponding to <sentence> elements. The words of each sentence are annotated as <token> children having up to 11 attributes:

- The value of the @id attribute identifies the word in the PROIELT.
- The value of the @form attribute corresponds to the actual word form in the text.
- The value of the @citation-part shows the reference of the token.
- The value of the @lemma is the dictionary form of the word.
- The @part-of-speech attribute contains the part of speech of the word.
- The value of the @morphology attribute is the morphological annotation of the word: it consists of a 10-long character string, where each position corresponds to an ordered value: e.g., the @morphology value of the word ἐγέννησεν is 3saia---i, which means that the word contains the following features: third person (3), singular (s), aorist (a), indicative (i), active (a), inflecting (i).
- The value of the @head-id attribute contains the @id value of the governor word.
- The value of the @relation attribute identifies the relation of the word with respect to its governor: e.g., the word Ναασσών is the object of the verb ἐγέννησεν (see Haug (2010) for the list and meaning of all relations).
- The value of the @antecedent-id attribute is the @id of the token with which a given token is coreferential.
- The attribute @information-status describes the cognitive state related to a word.
- The value of the @presentation-after attribute contains the punctuation mark that may follow the word. Note that in the AGDT punctuation marks are annotated not as attributes but as <word> elements.

3. Method

The treebanks have been searched for the number of verbs governing preverbal or postverbal accusative objects, which are nouns or pronouns, in the single or coordinate independent clauses of the following works: Homer's Iliad and Odyssey and the annotated parts of Herodotus' Histories and the New Testament.[5] To query the corpora I used XQuery 1.0 as implemented in BaseX 7.8.[6]

3.1. The query for the AGDT

The following is the query used to extract the data from the AGDT:

```
xquery version "1.0";
let $r :=
```

[5]The texts used are made available at https://github.com/gcelano/PBML101_2014.

[6] http://basex.org/products/download/all-downloads/.

```
for $t in //word[@relation=  ("PRED", "PRED_CO")]
let $o := $t/preceding-sibling::word[@relation = "OBJ"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head = $t/@id]
let $d := $t/parent::sentence/word[@relation= "COORD"]
        [@head = $t/@id]
let $y := $t/preceding-sibling::word[@relation= "OBJ_CO"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head = $d/@id]
let $h := $t/following-sibling::word[@relation = "OBJ"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head = $t/@id]
let $z := $t/following-sibling::word[@relation= "OBJ_CO"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head = $d/@id]
let $x := $t/parent::sentence/word[@relation=  "COORD"]
        [@id = $t/@head]
let $a := $t/preceding-sibling::word[@relation = "OBJ"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head=  $x/@id]
let $s := $t/following-sibling::word[@relation = "OBJ"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head=  $x/@id]
let $j := $t/parent::sentence/word[@relation= "COORD"]
        [@head = $x/@id]
let $b := $t/preceding-sibling::word[@relation = "OBJ_CO"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head=  $j/@id]
let $c := $t/following-sibling::word[@relation = "OBJ_CO"]
        [@postag[matches(.,"(n|p).....a.")]]
        [@head=  $j/@id]
(: the following where clauses can be changed thus:
where $o or $y or $a or $b
where $h or $z or $s or $c
where ($o and $h) or ($o and $z) or ($o and $s) or ($o and $c)
      or ($y and $h) or ($y and $z) or ($y and $s) or ($y and $c)
      or ($a and $h) or ($a and $z) or ($a and $s) or ($a and $c)
      or ($b and $h) or ($b and $z) or ($b and $s) or ($b and $c) :)
where $o or $y or $a or $b
return <r cid= "{$t/@cid}" v= "{$t/@form}" cite= "{$t/@cite}">
        <pre_object>{data($o/@form)}</pre_object>
        <passage>{data($o/parent::sentence/word/@form)}
        </passage>
        <pre_object_co>{data($y/@form)}</pre_object_co>
        <passage>{data($y/parent::sentence/word/@form)}
        </passage>
        <pre_obj_attached_to_coord>{data($a/@form)}</pre_obj_attached_to_coord>
        <passage>{data($a/parent::sentence/word/@form)}
        </passage>
        <pre_obj_co_attached_to_coord>{data($b/@form)}</pre_obj_co_attached_to_coord>
        <passage>{data($b/parent::sentence/word/@form)}
        </passage>
```

```
        <fol_object>{data($h/@form)}</fol_object>
        <passage>{data($h/parent::sentence/word/@form)}
        </passage>
        <fol_object_co>{data($z/@form)}</fol_object_co>
        <passage>{data($z/parent::sentence/word/@form)}
        </passage>
        <fol_obj_attached_to_coord>{data($s/@form)}</fol_obj_attached_to_coord>
        <passage>{data($s/parent::sentence/word/@form)}
        </passage>
        <fol_obj_co_attached_to_coord>{data($c/@form)}</fol_obj_co_attached_to_coord>
        <passage>{data($c/parent::sentence/word/@form)}
        </passage>
        </r>
for $k at $n in $r
return <f number= "{$n}">{$k}</f>
```

In the AGDT the verbs of independent clauses are the only verbs to receive the relation PRED: any other kind of verb is annotated differently. In our study we have disregarded the case of empty verbs, i.e., verbs which do not surface in the sentence but the annotator assumes to be implied. Since the objects I am looking for are nouns or pronouns in the accusative case, I have used a regular expression to match the desired values of the @postag attribute.

The AGDT identifies coordinate constituents by adding the _CO suffix to the value of the @relation attribute. This means that in order to capture coordinate verbs, one has to also take into account verbs having relation PRED_CO. Furthermore, since coordinate objects are annotated as direct dependents of a conjunction, with such a conjunction depending on the verb grammatically related to the coordinate objects, it is possible to link such coordinate objects to their predicate only bypassing the coordinate conjunction (see the let $d, let $y, and let $z clauses).

Single OBJs are returned by means of the let $o and let $h clauses. When there are two or more PREDs coordinated, there may be an object depending on both of them: these kinds of objects are meant to be captured by the let $a and let $s clauses. If there are coordinate objects that depend on coordinate predicates, they are identified by the let $b and let $c clauses. As specified in the comment within the query, one has to change the where clause to get the following results:

- If where $o or $y or $a or $b is added, the query returns verbs with single and coordinate preverbal objects.
- If where $h or $z or $s or $c is added, the query returns verbs with single and coordinate postverbal objects.
- If where ($o and $h) or ($o and $z) or ($o and $s) or ($o and $c) or ($y and $h) or ($y and $z) or ($y and $s) or ($y and $c) or ($a and $h) or ($a and $z) or ($a and $s) or ($a and $c) or ($b and $h) or ($b and $z) or ($b and $s) or ($b and $c) is added, the query returns PRED verbs having at least one preverbal and one postverbal object at the same time.

3.2. The query for the PROIELT

The query used to interrogate the PROIELT is the following:

```
xquery version "1.0";
let $a :=
for $t in //token[@relation = "pred"][@form]
let $i := $t/parent::sentence/token
        [@part-of-speech = "G-"]
        [@id = $t/@head-id]
let $n := $t/parent::sentence/token[@relation = "pred"]
        [@part-of-speech=  "C-"]
        [@id = $t/@head-id]
let $z := $t/parent::sentence/token
        [@part-of-speech = "G-"]
        [@id = $n/@head-id]
let $p := $t/parent::sentence/token[@relation=  "obj"]
        [@part-of-speech = "C-"]
        [@head-id = $t/@id]
let $e := $t/preceding-sibling::token[@relation= "obj"][@morphology[matches(., "......a...")]]
        [@part-of-speech = ("Pp", "Ne","Ps", "Pi","Pt", "Pk","Px", "Pc","Pd", "Nb")]
        [@head-id = $t/@id]
let $o := $t/preceding-sibling::token[@relation=  "obj"][@morphology[matches(., "......a...")]]
        [@part-of-speech = ("Pp", "Ne","Ps", "Pi","Pt", "Pk","Px", "Pc","Pd", "Nb")]
        [@head-id = $p/@id]
let $w := $t/following-sibling::token[@relation= "obj"][@morphology[matches(., "......a...")]]
        [@part-of-speech = ("Pp", "Ne","Ps", "Pi","Pt", "Pk","Px", "Pc","Pd", "Nb")]
        [@head-id = $t/@id]
let $k := $t/following-sibling::token[@relation=  "obj"][@morphology[matches(., "......a...")]]
        [@part-of-speech = ("Pp", "Ne","Ps", "Pi","Pt", "Pk","Px", "Pc","Pd", "Nb")]
        [@head-id = $p/@id]
(: the following two clauses serve to identify coordinate verbs sharing an object.
Change the direction of the greater than sign in all the following clauses at the same time:
> captures preverbal objects, while < captures postverbal objects :)
let $b := let $l := $t/parent::sentence/token[@relation = "pred"]
        [@head-id = $n/@id]/slash[@relation= "obj"][@target-id=  $e/@id]
        where $l/parent::token/xs:integer(@id) > $e/xs:integer(@id)
        return $l
let $c := let $v := $t/parent::sentence/token[@relation = "pred"]
        [@head-id = $n/@id]/slash[@relation= "obj"][@target-id=  $p/@id]
        where $v/parent::token/xs:integer(@id) > $o/xs:integer(@id)
        return $v
let $m := let $r := $t/parent::sentence/token[@relation = "pred"]
        [@head-id = $n/@id]/slash[@relation= "obj"][@target-id=  $w/@id]
        where $r/parent::token/xs:integer(@id) > $w/xs:integer(@id)
        return $r
let $f := let $y := $t/parent::sentence/token[@relation = "pred"]
        [@head-id = $n/@id]/slash[@relation= "obj"][@target-id=  $p/@id]
        where $y/parent::token/xs:integer(@id) > $k/xs:integer(@id)
        return $y
(: the following where clause can be changed thus:
where (not($i) and not($z)) and ($e or $o)
where (not($i) and not($z)) and ($w or $k)
```

```
where (not($i) and not($z)) and (($e and $w) or ($e and $k)
                                 or ($o and $w) or ($o and $k))
where (not($i) and not($z)) and ($b or $c or $m or $f) :)
where (not($i) and not($z)) and ($e or $o)
return
<r id= "{$t/@id}" locus= "{$t/@citation-part}" v= "{$t/@form}">
<pre_object>{data($e/@form)}</pre_object>
<passage>{data($e/parent::sentence/token/@form)}</passage>
<pre_object_co>{data($o/@form)}</pre_object_co>
<passage>{data($o/parent::sentence/token/@form)}</passage>
<fol_object>{data($w/@form)}</fol_object>
<passage>{data($w/parent::sentence/token/@form)}</passage>
<fol_object_co>{data($k/@form)}</fol_object_co>
<passage>{data($k/parent::sentence/token/@form)}</passage>
<object_of_co_verb1 v= "{$b/parent::token/@form}">{data($e/@form)}</object_of_co_verb1>
<passage>{data($b/parent::token/parent::sentence/token/@form)}</passage>
<object_of_co_verb2 v= "{$c/parent::token/@form}">{data($o/@form)}</object_of_co_verb2>
<passage>{data($c/parent::token/parent::sentence/token/@form)}</passage>
<object_of_co_verb3 v= "{$m/parent::token/@form}">{data($w/@form)}</object_of_co_verb3>
<passage>{data($m/parent::token/parent::sentence/token/@form)}</passage>
<object_of_co_verb4 v= "{$f/parent::token/@form}">{data($k/@form)}</object_of_co_verb4>
<passage>{data($f/parent::token/parent::sentence/token/@form)}</passage>
</r>
for $m at $x in $a
return
<r number= "{$x}">{$m}</r>
```

In the PROIELT the function of verbs in single or coordinate independent clauses is invariantly labeled as "pred"; the same label, however, is used to annotate the function of the verb of clauses introduced by a subordinate conjunction. This means that, in order to only capture single or coordinate independent clauses, one has to filter the number of predicates by using the where clause not($i) and not($z), which allows exclusion of predicates (not($i)) and coordinate predicates (not($z)) depending on a subordinate conjunction.

The predicate [@form] in the for $t clause serves to select only verbal predicates which appear in the texts (i.e., to exclude implied verbs, which are annotated as tokens having no @form attribute). The let $e and let $o clauses capture the single and coordinate preverbal objects, respectively: the predicates [@part-of-speech = ("Pp", "Ne", "Ps", "Pi", "Pt", "Pk", "Px", "Pc", "Pd", "Nb")] and [@morphology[matches(., ".....a...")]] are meant to select only nouns and pronouns in the accusative case. The let $w and let $k clauses are the counterparts of the let $e and let $o clauses, in that they concern following objects.

The where clause always contains the logical expression not($i) and not($z), which excludes the single and coordinate verbs of subordinate clauses. The same where clause has also to contain the expression and ($e or $o) if the single and coordinate preverbal objects are searched for, or the expression and ($w or $k) if the single and coordinate postverbal objects are searched for, or the expression and (($e and $w)

	preverbal objects	postverbal objects
Homer	4995	2451
Herodotus	804	753
the New Testament	997	2084

Table 1. Figures for preverbal and postverbal accusative object nouns and pronouns

or ($e and $k) or ($o and $w) or ($o and $k)), if the co-occurrences of preverbal and postverbal objects are searched for.

In the PROIELT, constituents relating to more than one finite verb are annotated differently than in the AGDT: one object is attached to a verb and, if it is also related to another coordinate verb, this verb contains a <slash> element pointing to the object (see Haug (2010) for a full description). This phenomenon is searched for through the let $b, let $c, let $m, and let $f clauses. More precisely, to find the preverbal objects of coordinate verbs, one has to use the greater than sign in all of the four clauses; to find postverbal objects, the less than sign is to be used.

The results of the where (not($i) and not($z)) and ($b or $c or $m or $f) clause are added to those obtained by applying the where (not($i) and not($z)) and ($e or $o) clause when the greater than sign is used; when the less than sign is used, the results are added to those returned by limiting the query with the where (not($i) and not($z)) and ($w or $k) clause. There are no coordinate objects with a <slash> element pointing to two objects, so there is no possibility for a coordinate verb to have both a preverbal and a postverbal object in the same clause.

4. Results

The results are presented in three tables. Table 1 contains the figures for verbs governing preverbal or postverbal objects, both single and coordinate; Table 2 contains the figures for verbs having preverbal and postverbal objects in the same clause; Table 3 shows the figures of the first table minus those of the second table. If we count two or more objects being on the same side of a verb as one object and the object relating to two coordinate verbs as two objects, we can say that the tables report the numbers of preverbal and posterbal objects.

5. Discussion

The results prove that in my files single or coordinate accusative object nouns or pronouns in single or coordinate independent clauses in Homeric poems are usually before the verb, while in the New Testament the same kind of objects are typically placed after the verb. Interestingly, Herodotus' text shows no real predominant order. This result is essentially in accordance with that found by Dunn (1988) (but he pro-

	co-occurrences
Homer	163
Herodotus	14
the New Testament	18

Table 2. Figures for co-occurrences of preverbal and postverbal accusative object nouns and pronouns

	preverbal objects	postverbal objects
Homer	4832	2288
Herodotus	790	739
the New Testament	979	2066

Table 3. Figures for non-co-occurring preverbal and postverbal accusative object nouns and pronouns

vides figures only for the first Book of the Histories, and it is not clear to me whether, as seems probable, he only analyzed single and coordinate independent clauses).

It is possible that further research will show that other kinds of texts of the Classical age had a marked preference for one order or the other, as occurs in Homer and the New Testament. Notwithstanding, my data prove that Classical Greek admitted at least a variety of the language (i.e., the literary one of historical texts) in which OV and VO orders are virtually equally licensed. Although the figures for the three texts clearly suggest a change from OV to VO order (which has always been assumed but never adequately justified), further texts will have to be analyzed to properly understand the conditions of this change, (e.g., how the genre of a text influenced word order).

The figures given do not explain why accusative object nouns and pronouns can be preverbal or postverbal: as it is known, this question can be successfully explored only in the light of studies on information structure: word order in AG is heavily determined by the categories of topic and focus and the phonology associated to them (this is prototypically shown by the distinction between nouns and pronouns; see Celano (2013a,b) and the bibliography therein). Notwithstanding, scientific quantification does prove to be a necessary means to guide research by exactly describing the language: the present analysis suggests that a word order change form OV to VO occurred over time.

6. Conclusions

In this article I have quantified the number of verbs with preverbal or postverbal object nouns or pronouns in single or coordinate independent clauses in Homer's Iliad and Odyssey, and the annotated parts of Herodotus' Histories and the New Testament. The results show that over time there was a shift from OV to VO order in AG. Although this has already been argued, the present research turns out to be the first attempt to quantify AG word order scientifically.

Dedication

Dedico questo articolo al Prof. Gregory Crane, difensore e promotore degli studi classici nel mondo.

Bibliography

Bamman, David and Gregory Crane. Guidelines for the syntactic annotation of the Ancient Greek Dependency Treebank (1.1), 2008. URL `http://nlp.perseus.tufts.edu/syntax/treebank/agdt/1.7/docs/guidelines.pdf`.

Celano, Giuseppe Giovanni Antonio. Argument-focus and predicate-focus structure in Ancient Greek: Word order and phonology. *Studies in Language*, 37(2):241–266, 2013a. URL `https://benjamins.com/#catalog/journals/sl.37.2.01cel/details`.

Celano, Giuseppe Giovanni Antonio. Word order. In *Encyclopedia of Ancient Greek language and linguistics*. Brill Online, 2013b. URL `http://brillonline.nl/entries/encyclopedia-of-ancient-greek-language-and-linguistics/word-order-EAGLL_COM_00000378?s.num=0&s.q=word+order`.

Cervin, Richard Stuart. *Word order in Ancient Greek: VSO, SVO, SOV, or all of the above?* PhD thesis, University of Illinois at Urbana-Champaign, 1990.

Dunn, Graham. Syntactic word order in Herodotean Greek. *Glotta*, 66:63–79, 1988.

Ebeling, H. L. Some statistics on the order of words in Greek. In *Studies in Honor of Basil L. Gildersleeve*, pages 229–240. Johns Hopkins University Press, Baltimore, 1902.

Friederich, Paul. Proto-Indo-European syntax: The order of meaningful elements. *Journal of Indo-European Studies*, 1, 1975.

Hajičová, Eva, Zdeněk Kirschner, and Petr Sgall. A manual for analytic layer annotation of the Prague Dependency Treebank (English translation), 1999. URL `http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/a-layer/html/index.html`.

Haug, Dag Trygve Truslew. Proiel guidelines for annotation, 2010. URL `http://folk.uio.no/daghaug/syntactic_guidelines.pdf`.

Kwong, Ivan Shing Chung. *The word order of the gospel of Luke: Its foregrounded messages*. T&T Clark, London, 2005.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel Publishing Company and Academia, Dordrecht & Prague, 1986.

Taylor, Ann. The change from SOV to SVO in Ancient Greek. *Language Variation and Change*, 6 (1):1–37, 1994.

Address for correspondence:
Giuseppe G. A. Celano
giuseppe.celano@tufts.edu
Tufts University
Perseus Project/Department of Classics
134C Eaton Hall
Medford, MA 02155, USA

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 2014 111–122

Productive verb prefixation patterns

Jaroslava Hlaváčová, Anna Nedoluzhko

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

Abstract

The paper discusses a set of verbal prefixes which, when added to a verb together with a reflexive morpheme, change the verb's meaning always in the same manner. The prefixes form a sequence according to the degree of intensity with which they modify the verbal action. We present the process of verb intensification in three Slavic languages, namely Czech, Slovak and Russian.

Motto[1]

Na záhrade sa rozťukal ďateľ.
Chvíľu si poťukal na jednom strome, potom si zaťukal aj na ďalších.
Naťukal sa za celý deň dosť.
Ak sa dnes nevyťukal, bude zajtra pokračovať.
Ale ak sa uťukal, tak nebude.

English translation:

A woodpecker started pecking in the garden.
He pecked for a while on one tree, then he did some more pecking on others.
During the whole day he pecked quite enough.
If he has not pecked enough, he will continue tomorrow.
However, if he tired himself out with pecking, he will not.

[1]We thank our colleague Jano Hric from MFF UK in Prague for composition of the mini-story in Slovak.

© 2014 PBML. All rights reserved. Corresponding author: hlavacova@ufal.mff.cuni.cz
Cite as: Jaroslava Hlaváčová, Anna Nedoluzhko. Productive verb prefixation patterns. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 111–122. doi: 10.2478/pralin-2014-0007.

1. Introduction

Unlike classic dictionaries, electronic ones are potentially unrestricted. They can include any number of words of a given language. Nevertheless, electronic dictionaries still cannot contain all the words. There are a lot of reasons why not all the words can be included in any dictionary. For example, there might be a disagreement among language speakers which words should be considered a part of the language and which should not. These are, at the first hand, foreign words, which in modern Slavic languages are mostly borrowed from English. Also, proper names are often problematic, especially the foreign ones. There exists a considerable discussion concerning neologisms, colloquial forms, casual words and so on.

A casual word is often constructed from an already existing word by adding an already existing prefix or suffix. These words are the subject of our investigation. In this paper, we will focus on imperfective verbs in Czech, Slovak and Russian, which have a rich potential of accepting prefixes.

In the three mentioned languages, we have found a set of prefixes that can be potentially added to almost all imperfective verbs. Being added to a verb together with a reflexive morpheme, each of these prefixes represents a modification of the verb original meaning. More specifically, this construction (prefix plus reflexivization) changes the intensity of the action. Some useful remarks on this topic can be found in Isachenko (1960). The Czech prefixes with the intensification meaning are mentioned also in Kopečný (1962).

According to the degree of intensity, these prefixal verbs can be organized in the similar way as the degrees of comparison for adjectives or adverbs. Therefore, although by a certain stretch, such kind of prefixation (plus reflexivization) may be called "verb intensification".

The analysis of verb intensification will be proposed for three Slavic languages: Czech, Slovak and Russian. However, we assume that this derivation pattern might hold for other Slavic languages as well.

2. Intensifying prefixes

Majority of Czech, Slovak and Russian imperfective verbs might be "intensified" according to a simple rule:

1. add a prefix from a special closed set of prefixes and
2. add a reflexive morpheme, namely the reflexive particle *se*, *si* in Czech, *sa*, *si* in Slovak, and the reflexive suffix *-ся* (*-сь*) in Russian.

Prefixes to be discussed are summarized in Table 1, where they are sorted according to an intensity they modify the verbal action expressed by the basic verb. For easier referencing in the following text, we assigned initial letters to these prefixes (in Latin alphabet). If we mention, for instance, the prefix Z-, we will have in mind all the prefixes in the column superscribed Z- in Table 1, namely *za-* in Czech and Slovak and *за-* in Russian.

Abbreviation	R	P	Z	N	V	U	I	D
Czech	roz-/roze-	po-	za-	na-	vy-	u-		
Slovak	roz-/rozo-	po-	za-	na-	vy-	u-		
Russian	раз-/рас-/разо-	по-	за-	на-	вы-	у-	из-/ис-/изо-	до-

Table 1. Prefixes of intensification for Czech, Slovak and Russian

We can see that the row containing the Russian prefixes is extended by two columns. It contains the additional prefixes *из-/ис-* and *до-*, which represent the highest degrees of intensity. Although these prefixes also exist in Czech and Slovak, they do not have the intensified meaning.

Table 2 characterizes briefly the individual prefixes, especially their intensification role in Czech and Slovak, Table 3 presents the same for Russian.

Prefix	Verb	Reflexive morpheme	Meaning
R-	X	*se/sa*	start to X
P-	X	*si*/se/sa*	X calmly, mostly in pleasant manner
Z-	X	*si*/se/sa*	X taking some time, enjoying it
N-	X	*se/sa*	X intensively
V-	X	*se/sa*	X intensively, with a satisfaction
U-	X	*se/sa*	X to exhaustion

Table 2. Survey of intensifying Czech and Slovak verbal prefixes. The asterisk (*) in the table means that the morpheme si *is employed for originally non-reflexive verbs. In the case of reflexivum tantum, the morpheme* se/sa *remains.*

Dividing the information into two tables is intentional. We want to point out the sameness of the Czech and Slovak languages, and their differences in comparison to Russian. They concern not only different meanings of some prefixes, but also the reflexivity of resulting verbs, especially of those related

Prefix	Verb	Refl. morpheme	Meaning
R-	X	*-ся*	start to X or strengthen the intensity of X
P-	X	*0/себе*	X during a relatively short time period
Z-	X	*-ся*	immerse into X for a longer time period
N-	X	*-ся*	exhaustively X, often with a positive result
V-	X	*-ся*	X until exhaustion
U-	X	*-ся*	X until exhaustion
I-	X	*-ся*	X until total exhaustion, often with a neg. result
D-	X	*-ся*	X with high intensity leading to a neg. result

Table 3. Survey of intensifying Russian prefixes.

to prefixes P- and Z-. We will demonstrate the differences on examples in the following section.

Although the intensifying meaning of the individual prefixes is always the same, their intensity is not absolute in neither of the languages. The meanings overlap. Only the prefix R- might be assigned an absolutely lowest intensity in Czech and Slovak. In both languages, it represents beginning of an action, thus the weakest intensity. On the contrary, the prefix U- represents the highest intensity.

In Russian, the overlapping seems to affect all the prefixes, including the extreme ones. Here, we might spot another ordering, namely according to a positive or negative result of the verbal action. We present two pictures (1 and 2) demonstrating the sequence of the prefixes according to their (possibly overlapping) intensity (see also similar pictures in Hlaváčová and Nedolužko (2012) and Hlaváčová and Nedoluzhko (2013)). In Russian, the intensity of the last five prefixes is very similar (see the the right end of Figure 2).

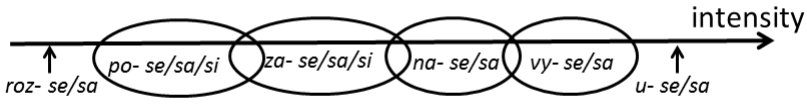

Figure 1. Czech intensifying prefixes

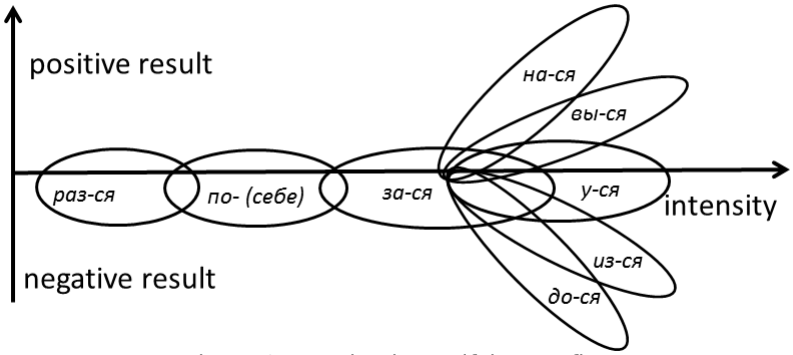

Figure 2. Russian intensifying prefixes

3. Examples

In what follows, we present a set of examples for all studied prefixes in each of languages mentioned above. Every example consists of one sentence in all the languages. The sentences are not authentic, as it would be really very difficult to find the same sentence in three languages, especially for such a rare phenomenon. Even so, for some prefixes it was not possible to translate the same sentence into all three languages by means of verb intensification. The Czech and the Slovak languages are very similar, so translations between them was easy but the meaning of several Russian prefixes is so different that some sentences had to be reformulated. Otherwise the translated sentence would sound unnatural. The prefix Z- varies in Russian so much that it was impossible to use the same prefix for all three translations. In that case we had to come up with a totally different sentence to demonstrate the prefix meaning.

There are authentic examples from the Czech National Corpus SYN (CNK) cited in Hlaváčová (2009). The works Hlaváčová and Nedolužko (2012) and Hlaváčová and Nedoluzhko (2013) contain more examples in Czech and in Russian. The latter were found in the Russian National Corpus (RNC) and on the Internet.

Prefix R-

Czech	*Před vystoupením se rozezpíval.*
Slovak	*Pred vystúpením sa rozospieval.*
Russian	*Перед выступлением он распелся.*
In English	*To be prepared, he started to sing before the performance.*

Prefix P-

Czech	*Pohrál si s nastavením parametrů.*
Slovak	*Pohral si s nastavením parametrov.*
Russian	*Он немного поиграл с настройкой параметров.*
In English	*He played a bit with the parameter settings.*

Prefix Z-

Czech	*Zatrénoval si v novém dresu, starý už byl roztrhaný.*
Slovak	*Zatrénoval si v novom drese, starý už bol roztrhaný.*
In English	*He enjoyed training in a new sports dress, the old one had been torn.*
Russian	*Он заработался и забыл про обед.*
In English	*He buried himself to work and forgot to have a lunch.*

Prefix N-

Czech	*Po vystoupení se naděkoval víc než minule.*
Slovak	*Po vystúpení sa naďakoval viac než minule.*
Russian	*После выступления он никак не мог наблагодариться.*
In English	*After the performance, he aknowledged the applaus more than usually.*

Prefix V-

Czech	*Punťa se na zahradě vyběhal do sytosti.*
Slovak	*Dunčo sa na záhrade vybehal do sýtosti.*
Russian	*Шарик выбегался досыта в саду.*
In English	*The dog ran in the garden as much as it liked.*

Prefix U-

Czech	*Na komedii se uchechtal až k slzám.*
Slovak	*Na komédii sa uchechtal až k slzám.*
Russian	*На комедии он ухохотался до слез.*
In English	*On the comedy, he roared with laughter, almost to tears.*

Prefix I-

Russian	*Этот автор уже совсем исписался.*
In English	*This author has exhausted all topics and he has nothing to write about.*

Prefix D-

Russian	*Первоклашка добегался до перелома ноги.*
In English	*The first-grader ran around so much that he ended up with a broken leg.*

4. Intensified verbs in dictionaries and homonymy

The examples above show that the meaning of the prefixes is quite clear and it changes the meaning of a basic verb always in the same way. Some prefixal verbs have been already adopted in the language, becoming part of the common vocabulary. One of such words is the verb *rozesmát se – rozosmiať sa – рассмеяться* (*to start laughing*).

However, in many cases, the prefixal verbs of such kind are uncommon. They are used only occasionally and that is why they cannot be found in dictionaries. This is the case of most verbs with the basis *ťukať* (*to peck* in English) from the Slovak motto in the beginning of this paper. It is certainly possible to paraphrase this mini-story without using the intensified verbs. However, using this kind of verbs enables to express the story in a better, more colorful and witty way, and this is the reason why such casual words appear, though not very often. For humans, it is easy to understand such verbs, though they do not appear in any dictionary. It is sufficient to understand the prefix, the verb and its reflexivization.

With the means of the verb intensification, there can be created verbs that already exist in a common vocabulary of a language but have a different meaning. Such verbs are homonymous. An example is the Czech verb *usmát se*. In The Dictionary of Contemporary Czech (SSJC), this word is explained as "express gladness by a smile". This meaning differs essentially from the intensified meaning "to be totally exhausted by laughing" formed by prefixation + reflexivization. Compare the following Czech examples:

Usmála se na svého přítele.
(*She smiled at her friend.*)

Smála se tak dlouho, až se téměř usmála.
(*She laughed for such a long time that she was completely worn out by laughing.*)

Unlike academic dictionaries of Czech and Slovak (e.g., SSJC; SSJ), the Dictionary of Russian (MAS 1999) contains many uncommon intensification meanings. For example, for the verb *распишаться*, the dictionary gives both common meanings (*sign one's name* and *register one's marriage*) and a relatively rare intensifying meaning (*get into a writing vein*). However, it does not include intensification verbs systematically, for example, the saturative meaning of *наплаваться* (*to swim a lot*) is presented, but not *наныряться* (*to dive a lot*).

5. Verb intensification as an inflectional morphological category

The question is how to treat intensified verbs within automatic language processing, namely how to lemmatize them. It is commonly accepted that a lemma has the same prefix as all of the word-forms which can be derived from it. The exception for the Czech and Slovak are the negation prefix *ne-* and superlative intensifying prefixes *nej-/naj-/най-* of adjectives and adverbs in all the analyzed Slavic languages. This gave us an idea of using the notion "intensification" for regular composing prefixal reflexive verbs with the listed set of prefixes. Taking this into account, it makes sense to use the infinitive of unprefixed verbs as the lemma of the intensified verbs. For example, the Czech intensified verb *vyběhal se* with the prefix V- (see Section 3) will be lemmatized as *běhat* (*to run*).

This decision seems to be more precise than using the lemma with the prefix, because this will emphasize that the prefixal word-form belongs to the paradigm of the basic unprefixed verb. Indeed, the prefix does not change the meaning of the verb in these cases, it only modifies its intensity.

Another advantage of such decision is the possibility to resolve the homonymy of the forms mentioned in 4. In the first example, the word-form *usmála* will be assigned the lemma *usmát (se)* (*to smile*) [2]. In the second case, the lemma will be *smát (se)* (*to laugh*). With such a lemmatization, the information about the degree of intensification can be observed as an inflectional morphological category for verbs. This category can be called intensification, similarly as degrees of comparison for adverbs and adjectives. This morphological category can be realized with the means of the affixes in question together with a reflexive morpheme of a given language.

Thus, we believe that adding one of the listed prefixes and a reflexive morpheme (with the exception of the prefix P- in Russian — see Table 3) to a verb does not create a new verb but a new form of the same verb.

6. Automatic recognition of intensified verbs in texts

Morphological analysis assigns morphological characteristics and lemmas to each word. For this purpose, extensive morphological dictionaries, including all[3] words of the given languages, are used. Such dictionaries exist for

[2] The question of the reflexive morpheme being a part of the lemma is not discussed in this paper.

[3] as many as possible

all three languages under analysis. However, real texts still include unknown words. For recognition of such words, so called guessers are usually used. They are able to guess the morphological characteristics of unknown (out of vocabulary, or OOV) words. The guesser should be able to distinguish intensified verbs and to assign them an appropriate set of morphological properties, including a lemma.

First of all, intensified verbs should be recognized in a text. If they are included in the dictionary, they will be analyzed in the same way as other words. In that case, there is no reference to their intensified nature, because dictionaries often do not include this sort of information. This concerns common verbs that have already been lexicalized.

For an unknown word, there is quite easy procedure how to check if a verb in question is the result of the intensification. The following two conditions must be fulfilled:

- The word can be split into two strings AB, where A belongs to the set of intensification prefixes, B is an imperfective verb (recognized as such by the morphological analysis).
- There is a reflexive morpheme either attached to the verb (in case of Russian, with the exception of the prefix P) or within a certain span around the word in the same sentence (in case of Czech and Slovak).

Under these conditions, there is a strong probability that the unknown word might be considered to be an intensified verb. We have carried out a pilot experiment for recognition of intensified verbs in the Czech national corpus SYN2000 (CNK). We extracted all unknown words starting with given prefixes that are immediately followed by the reflexive *se* or *si*. We ignored sentences with the reflexive before the word, as it would be more difficult to distinguish, whether the reflexive belongs really to the word. Then, we removed the prefix and analyzed the rest of the word. If the rest did not appear to be an imperfective verb[4], we did not take it into account. The resulting sets of candidates have been manually investigated and the intensified verbs have been collected. The number of candidates and real intensified verbs for each prefix is presented in Table 4.[5]

[4]We also excluded transgressives from the set, as they are very rare, but homonymous with other parts of speech.

[5] For such kind of results, it is common to calculate precision, but due to small amount of data it is not statistically significant.

Prefix	Candidates	Real intensified verbs
roz-	25	20
po-	11	5
za-	6	4
na-	11	2
vy-	2	0
u-	7	2

Table 4. Number of candidates for intensified verbs.

The figures presented in Table 4 show, that the rule for the recognition of intensified verbs is not perfect. However, the false findings were usually due to various errors in the original texts. For example, there are typos such as the word *postekl*, where *stekl* has the lemma *stéci — to flow down*, instead of the correct *posteskl* (*to express dissatisfaction*). Also, words used in a rather unusual form which is missing in the dictionary are problematic, e.g., the verb *vyhlo* as a variant of *vyhnulo* (*avoided*).

Another evidence that could help recognizing intensified verbs in a sentence is the use of certain intensifiers. For example, the typical intensifiers for the prefix U- are *do sytosti, dosyta* (to one's heart's content), *úplně* (completely), *k smrti* (to death), *do (úplného) vyčerpání* (to (complete) exhaustion) in Czech, *dosýta, úplne, k smrti, do (úplného) vyčerpania, do sýtosti* with the same meanings in Slovak, and *совсем* (entirely, totally), *досыта* (to one's heart's content), *до смерти* (to death) in Russian.

The information about the verb intensification can be useful in some applications of automatic language processing, e.g., in machine translation. If the input and the output languages have similar intensification degrees, the same principles for words in both languages can be applied. For example, the Slovak word *rozťukať sa* (*to start pecking*) from the Motto will be translated into Czech as *rozťukat se*, although this verb possibly does not exist in any morphological dictionary of either of these languages. In the languages without intensification, for example in English, the special rules for translating this prefix should be applied. For example, the prefix R- is likely to be translated using the expression *start to*.

In automatic language processing, it is also important to recognize the homonymy between basic and intensified verbs (see the example above with the Czech verb *usmát se — smile* or *to be completely exhausted by laughing*). Such

a distinction seems to be difficult at first sight, but there are certain signs that help revealing intensified verbs in a sentence. One of such signs is the fact that intensified verbs typically lack an object, even in case of transitive verbs. Once the verb is intensified, its object is generalized and cannot be explicitly expressed in the sentence. Cf. *číst knihu* (*read the book*) — *učíst se k smrti* (*to read to death*), but not **učíst se knihu k smrti* (*to read the book to death*). The same can be claimed about most other verbs that include objects or adverbials in their valency frames. The deeper analysis of syntactic and semantic constraints of the intensification pattern in Russian is provided in Khoroshkina and Nedoluzhko (2014).

7. Conclusion

We presented a set of productive word formation patterns composed of a prefix and a reflexive morpheme. Using the examples of Czech, Slovak and Russian, we have shown that this procedure can be applied to almost any imperfective verb in the above mentioned languages. This does not change the basic meaning of the verb, but only the intensity of the action which is expressed by this verb. The intensification resembles the degrees of comparison for adverbs and adjectives. Inspired by this similarity, we have called it "verb intensification".

The presented approach implies treating the verb intensification differently from the current praxis. We propose to lemmatize newly created verbs with an unprefixed infinitive, thus including the form of intensification into the paradigm of the unprefixed verb.

As for semantic characteristics, the frequency of occurrences of the intensification patterns is higher with the verbs that have an animate actor, who is able to control the action or at least to perceive it. Therefore, intensified verbs such as, e.g., *nalyžovat se* (*to ski one's fill*) and *naradovat se* (*enjoy oneself to the fullest*) sound much more natural than *napraskat se* (*to crack one's fill*) or *naslyšet se* (*to hear one's fill*), though they are not completely impossible. The same applies to the semantic feature of iterativeness: the action has to be "repeatable" to be able to take the intensification pattern, cf. *?naumírat se*, which is only compatible with the context of computer games, where dying can be considered a repeatable action. However, the more detailed analysis of semantic constraints of this kind should be provided for all analyzed languages.

Furthermore, the word-formative patterns of different groups of verbs are of different degree of productivity. More than that, the prefixes themselves are

not equally productive in different analyzed languages. One could possibly assume that in Czech and Slovak, the verb intensification might be more common than in Russian. However, this claim should be proved by a more specific analysis of the use of intensified verbs in these languages. Also the acceptability of certain examples is sometimes ambiguous and requires a broader context. A further research should be performed to bring clarity to these issues.

Bibliography

CNK. Czech national corpus, 2012. <http://www.korpus.cz>.

Hlaváčová, Jaroslava. *Formalizace systému české morfologie s ohledem na automatické zpracování českých textů*. PhD thesis, FF UK, Nám. Jana Palacha 2, 116 38 Praha 1, 2009.

Hlaváčová, Jaroslava and Anna Nedoluzhko. Intensifying verb prefix patterns in czech and russian. In *TSD 2013. Proceedings*, volume 8082 of *LNCS*, pages 303–310, Berlin / Heidelberg, 2013. Západočeská univerzita v Plzni, Springer Verlag. ISBN 978-3-642-40584-6.

Hlaváčová, Jaroslava and Anja Nedolužko. Příklad pravidelných slovotvorných vzorců v automatickém zpracování češtiny a ruštiny. In Horváth, Tomáš, editor, *Zborník príspevkov prezentovaných na konferencii Informačné technológie – Aplikácie a Teória, ITAT*, pages 53–56, Košice, Slovakia, 2012. Univerzita Pavla Jozefa Šafárika v Košiciach. ISBN 978-80-971144-1-1.

Isachenko, A.V. *Grammar of Russian language in comparison with the Slovak*, volume V.II. Jazyki slavjanskoj kultury, Bratislava, 1960. Grammaticheskii stroi russkogo yazyka v sopostavlenii s slovackim.

Khoroshkina, Anna and Anna Nedoluzhko. "Vchera nasochinyalsya voroh strok": Productive circumfixal intensifying patterns in Russian. In *Papers from the Annual International Conference Dialogue 2014*, volume 13 (20) of *Computational Linguistics and Intellectual Technologies*, Moscow, 2014.

Kopečný, František. *Slovesný vid v češtině*. Nakladatelství ČSAV, Praha, 1962.

MAS 1999. *Malyj Akademicheskii Slovar russkogo yazyka*. Poligrafresursy, Moscow, 1999. The Small Academic Dictionary of Russian.

RNC. Russian national corpus, 2011. <www.ruscorpora.ru>.

SSJ. *Slovník slovenského jazyka*. Bratislava: Vydavateľstvo SAV, 1959–1968.

SSJC. *Slovník spisovného jazyka českého*. Praha, Academia, 1989.

Address for correspondence:
Jaroslava Hlaváčová
hlavacova@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 101 APRIL 2014 123–146

A case of collision in principles of language description?

Roland Wagner

Masarykova univerzita, Czechia

Abstract

In the article, I deal with an apparent case of incoherence within the framework of Functional Generative Description (FGD) in treating certain reflexive constructions. According to the principle that differences between valency frames are constitutive for establishing different lexical units (e.g., Žabokrtský and Lopatková, 2007), the occurrences of *vnímat* in (i) *Vnímá syna jako soka* and (ii) *Sám se vnímá jako síla ochraňující divadlo* have to be assigned to two different lexical units, as there is a difference in the formal marking of the Effect-argument (*jako*+accusative case in (i), *jako*+nominative case in (ii)). On the basis of the commutation test (e.g., Panevová, 2008) which identifies the reflexive clitic *se* in (ii) as an object-clitic, both occurrences of *vnímat* have to be, however, assigned to the same lexical unit as the only difference between *vnímá syna* in (i) and *vnímá se* in (ii) consists in the lexical filling of the object position. In the main part of the text, I review two different strategies to remedy the conflict between the two principles. I point out certain implications of both strategies and conclude by pleading for a solution within the framework of Modified Valency Theory (Karlík, 2000a).[1]

1. Introduction

One of the advantages of the framework of Functional Generative Description (FGD) as it is developed since the sixties by a team of linguists associated with the Charles University in Prague (cf., e.g., Sgall et al., 1986) is the fact that it is systematically applied to a large corpus of linguistic data. By analyzing huge amounts of

[1] Some of the observations and analyses, especially those concerning the existence of the dative-type clitic *si* in Czech and its implications for the grammar of Czech, stem from my unpublished dissertation accepted in 2012 at the Department of Czech language at Masaryk University in Brno/Czech Republic (accessible from `http://is.muni.cz/th/56270/ff_d/wagner_prognoza_reflex.pdf`). The broader line of argumentation, however, is original to the present article.

© 2014 PBML. All rights reserved. Corresponding author: wagner@ped.muni.cz
Cite as: Roland Wagner. A case of collision in principles of language description?. The Prague Bulletin of Mathematical Linguistics No. 101, 2014, pp. 123–146. doi: 10.2478/pralin-2014-0008.

natural language data, lacunas or inherent contradictions in the descriptive apparatus inevitably become apparent and can be amended (Panevová, 1998). The present article deals with a conflict between two guiding principles of language description currently applied within FGD which seems to have gone previously unnoticed. One principle concerns the definition of the verbal lexical unit on the basis of its valency frame; the second principle is used to determine the status of the clitic reflexive *se* in different syntactic contexts. As will be shown in detail in the following text, both principles give inherently incoherent results when applied to Czech constructions containing, apart from further arguments, an Effect argument (EFF).

In Section 2, I try to formulate both principles under consideration as precisely as possible. Section 3 introduces the construction which is problematic with respect to these two principles of language description. All examples are based on the Czech verb *vnímat* 'to see, to regard (as)'. Further examples taken from the Valency Lexicon of Czech Verbs [VALLEX] (Lopatková et al., 2008) are given in Section 4. In Section 5, I provide a detailed discussion of two possible strategies how the collision of principles described in the previous sections can be avoided. One strategy is based on a suggestion made by Oliva (2001), which would allow eliminating the type of reflexives causing the problem from the descriptive apparatus altogether. The second strategy considered in Section 5 adopts Karlík's (2000a; 2000b) modification of Valency Theory for defining valency frames underspecified with respect to the formal features of certain arguments. As a consequence, all occurrences of the verbs which at present are problematic for FGP can be integrated into one lexical unit. Although the discussion leads to no definite conclusion, it hopefully highlights the problem and clarifies the implications which different solutions might have for the wider architecture of the grammar.

2. Two descriptive principles in FGD

FGD is a valency based approach to syntactic analysis. Each verb is said to be associated in the lexicon with a certain valency frame which encodes the number and the morphosyntactic form of the arguments to be realized in the syntax, cf. e.g., Žabokrtský and Lopatková (2007). In contrast to other linguistic theories which also rely in one way or the other on valency information (e.g., Chomsky, 1981; Wunderlich, 1997; Karlík, 2000a,b), no distinction is drawn between structural cases, which are assigned to their arguments in certain structural contexts and which are therefore entirely predictable, and idiosyncratic or "lexical" cases, which are unpredictable and have, therefore, to be listed together with the respective arguments in the lexical entry of the verb.

In accordance with the current practice in FGP, the Valency Lexicon of Czech Verbs [VALLEX] gives for every single argument of a lexical unit the morphological case and/or the preposition (e.g., *na*+4) which is to be associated with this argument in the syntax (cf. Žabokrtský and Lopatková, 2007, pp. 49–50). This approach necessar-

ily implies that the morphosyntactic information associated with the slots in a FGD valency frame has to be considered as an integral part of the valency frame. Two valency frames implying different morphosyntactic features for their arguments have to be considered as different frames, even if they coincide otherwise in the number and functional role type of the implied arguments.

A case at hand are the verbs *oslavovat / oslavit* 'to celebrate', *dosahovat / dosáhnout* 'to reach' and *patřit* 'to belong to'. According to [VALLEX], pp. 142, 60 and 149, respectively, all three verbs are associated with a two place valency frame (neglecting free modifications) including an Actor (ACT) and a Patient (PAT) argument. However, the PAT differs as to the morphological case of the nominal realizing the patient slot in the syntax. While the PAT of *oslavovat / oslavit* must be realized by a nominal in the accusative case (*oslavit své narozeniny* 'to celebrate one's birthday'), the PAT of the other two verbs has to be realized by a nominal in the genitive case (*nemoc dosáhla pokročilého stadia* 'the disease reached a progressed stage') and the dative case (*kniha patří Janovi* 'the book belongs to Jan'), respectively (examples taken from [VALLEX]). To guarantee the realization of the appropriate case form in the syntax, the Patient arguments of the three lexical entries are each marked for a different case feature (4 for accusative, 2 for genitive, 3 for dative case). The valency frames thus look as follows (again neglecting insubstantial details): ACT_1, PAT_4 for *oslavovat / oslavit*, ACT_1, PAT_2 for *dosahovat / dosáhnout* and ACT_1, PAT_3 for *patřit*. As the case features included in these valency frames convey distinctive information specific for the respective verbs, it follows, that $ACT_1, PAT_4 \times ACT_1, PAT_2 \times ACT_1, PAT_3$ are three different frames, even though they all include the same syntactic-functional roles (ACT and PAT).

In order to formulate the first principle of FGP as I see it, we now have to take a closer look at the relation between lexical entries for verbal lexemes and valency frames. Lexical entries in [VALLEX] are organized on two levels of abstraction. On a higher, more abstract level, the lexicon is organized in "lexemes", i.e. abstract lexical types defined on the basis of the phonological and morphological identity of the related tokens. Each "lexeme" may be further sub-divided into several "lexical units" (Žabokrtský and Lopatková, 2007) or "lexes" (Panevová, 1998). According to Žabokrtský and Lopatková (2007, p. 43), a lexical unit "corresponds [...] to the lexeme used in a specific sense and with specific syntactic combinatorial potential." As the valency frame reflects the combinatorial potential of the word form, it is associated not with the lexeme as a whole but with the lexical unit. Each lexical unit is therefore characterized by (a) a specific meaning, (b) a specific combinatorial potential reflected in its valency frame. Both aspects are constitutive for the lexical unit. It is, therefore, not feasible that one and the same lexical unit is associated with several different valency frames, even though a "lexeme" (as an entity defined on purely formal grounds) may be composed of different lexical units differing with respect to their valency frames.

Based on these considerations and following Panevová (1998), the first principle of FGP discussed in the present article can be formulated as follows:

Principle 1: Differences in valency frames correlate with differences in lexical meaning; to each lexical unit one and only one valency frame can be assigned.[2]

We now turn to the second principle which, as I claim in the present article, conflicts in certain cases with Principle 1. Principle 2 states how to treat verb forms co-occurring with reflexives. Within FGP (e.g., Panevová, 2008), three structural types of reflexive constructions are distinguished:

(a) the reflexive occurs either obligatory with the respective verb or indicates a difference in lexical meaning between the verb under discussion and a further, homonymous verb not co-occurring with a reflexive;
(b) the reflexive co-occurs with certain verb forms of an otherwise non reflexive verb indicating a non canonical mapping[3] of arguments onto surface syntactic positions;
(c) the reflexive co-occurs with a verb form of a non reflexive verb indicating co-reference of two argument positions.

In case (b), the reflexive is analyzed as a grammatical marker of a certain derived diathesis of the Czech verb called occasionally "reflexive passive". Reflexives in case (c) are analyzed as reflexive pronouns realizing the argument associated with the syntactic position they are occupying. Czech has a pronominal system distinguishing clitic and non-clitic forms, therefore reflexives of type (c) can appear in the clitic (*se* in the accusative, *si* in the dative case) as well as in the non-clitic form (*sebe* in the accusative, *sobě* in the dative case). As only reflexives of type (c) are analyzed as reflexive pronouns, the substitutability of the form *se*, resp. *si*, for *sebe*, resp. *sobě*, can be used as an operational test to distinguish reflexives of type (c) from reflexives of all other types. As to case (a), the reflexives obligatorily appearing with certain verbs are analyzed as an integral part of the respective lexical unit. This is so, because they are either a permanent feature of all occurrences of a certain verb (and have therefore to be lexically associated with the respective lexical unit) or distinguish different homonymous surface forms belonging to different lexical units (and are therefore a relevant part of the word structure of the respective units). As to their exact syntactic status, the theory is agnostic. Usually, they are referred to simply as "particles" (cf. Panevová, 1999, p. 271; 2008, p. 154; Žabokrtský and Lopatková, 2007, p. 44).

Taking for granted the analysis of reflexives reviewed above, I now formulate the second descriptive principle of FGP to be discussed in the present article:

[2] Panevová (1998, p. 2, footnote 4) concedes that this principle is, for practical reasons, not always respected in lexicographic work. For the user of dictionaries, it is sometimes convenient to unite lexical units with different frames into one entry. In this article, I retain the more rigorous version given above, as I am concerned with principle questions of linguistic description, not with possible applications for practical purposes.

[3] The term "non canonical mapping" is not used in the work cited above. It is introduced here to capture the difference between valency potency and valency realization, cf. Daneš et al. (1987); Ágel and Fischer (2010). The details are not relevant to the present discussion.

Principle 2: A reflexive which is not the clitic form of a reflexive pronoun or the marker of the grammatical category 'reflexive passive' is an integral part of the lexical unit; reflexive and non-reflexive verbs are two different lexical units.

Principle 2 has a corollary with regard to reflexives analyzed as reflexive pronouns. This corollary has to be made explicit, as it will play an important role in the following discussion.

Corollary of Principle 2: A reflexive element, which ***is*** the clitic form of a reflexive pronoun, is not part of the lexical unit, i.e.: A verbal form co-occurring with a (clitic) reflexive pronoun and a verbal form not co-occurring with a (clitic) reflexive pronoun belong ***to the same lexical unit*** (as long as other criteria for lexical identity are met).

We are now in a position to approach the collision between Principle 1 and Principle 2.

3. The case of collision

In order to see the conflict arising between the two descriptive principles of FGP formulated in the previous section, we now turn to a specific case of reflexivization. The point can be made most clearly by considering trivalent verbs including an Effect argument (EFF) in their valency frame. An EFF is "[…] the predicative complement (with such verbs as *elect*, *nominate*, *promote*) and the traditional adverbials of result in *he tore it to pieces* […]" Sgall et al. (1986, p. 134). A typical example of a verb including EFF in its valency frame is the verb *vnímat* 'to see, to regard, to consider (as)'. In (1), the third lexical unit within the lexeme *vnímat* is given:[4]

(1) Lemma: $vnímat_3$
Meaning: to see, to regard, to consider (as)
Valency frame: ACT_1, $PAT_{4,\ že}$, EFF_{jako+4}.

The morphosyntactic information added to PAT requires the Patient argument to be realized either as a nominal in the accusative case or as an embedded clause introduced by the subordinating conjunction *že* 'that'. The morphosyntactic information added to EFF states that the Effect argument has to be realized as a nominal in the accusative case introduced by the element *jako*. A larger structure containing the verb *vnímat* and its syntactic arguments is given in (2):

(2) Vnímá syna jako soka. (VALLEX, Lopatková et al., 2008, p. 295)
'He sees his son as a rival.'

[4] The entry is given according to [VALLEX], p. 295. The English paraphrase has been added.

The nominal *sok* 'rival' in (2) is in the accusative case, exactly as required by the valency frame. Now, let us take a look at another example showing the verb *vnímat* co-occurring with a reflexive. The example is taken from the Czech National Corpus:[5]

(3) Sám se vnímá jako „síla ochraňující divadlo". (SYN2005)
'He sees himself as a force sheltering theatre.'

We observe that the requirements of the valency frame given in (1) are not met in (3): Instead of being in the accusative case, the Effect argument *síla* 'force' appears in the nominative case (even though introduced by *jako*, as it is required by the lexical entry). Applying Principle 1 of FGP, we are forced to conclude that the verb form appearing in (2) and the verb form appearing in (3) belong to two different lexical units. Let us say, then, that sentence (2) exemplifies the lexical unit 'vnímat$_{\mathrm{I}}$' (including the argument EFF$_{\mathrm{jako+4}}$ in its valency frame) and that sentence (3) exemplifies the lexical unit 'vnímat$_{\mathrm{II}}$' (including the argument EFF$_{\mathrm{jako+1}}$ in its valency frame).[6]

The verb 'vnímat$_{\mathrm{II}}$' is co-occurring with a reflexive which is missing in the environment of 'vnímat$_{\mathrm{I}}$'. Apparently, the reflexive, then, is the formal marker of the lexical difference between both lexical units. Applying the classification of reflexives introduced in Section 2, we have to assign the reflexive in (3) to class (a): It is an integral part of the lexical unit indicating a lexical difference between the reflexive verb 'vnímat$_{\mathrm{II}}$' and the non-reflexive verb 'vnímat$_{\mathrm{I}}$'. We can now refer to the reflexive lexical unit 'vnímat$_{\mathrm{II}}$' by adding the reflexive to the lemma ('vnímat se') and reserve the lemma 'vnímat' without index to the basic lexical unit 'vnímat$_{\mathrm{I}}$'.

Now, consider example (4) containing a certain form of the verb 'vnímat' in one further context.

(4) Karel IV. vnímá sebe jako vyvoleného třetího krále. (SYN2005)
'Charles IV sees himself as the chosen third king.'

The sentence in (4) contains the non-clitic, "strong" pronominal form *sebe* which according to FGP forms a paradigm with the clitic reflexive *se* seen in (3). The sentences (3) and (4) can be therefore interpreted as minimal contexts for applying the substitution test mentioned in Section 2. Put differently, sentence (4) demonstrates that the

[5] The Czech National Corpus is accessible from `http://ucnk.ff.cuni.cz/`.

[6] In order to avoid misunderstandings, it has to be emphasized that the hypothetical lexical units 'vnímat$_{\mathrm{I}}$' and 'vnímat$_{\mathrm{II}}$' are not given in [VALLEX]. They are introduced at this place by the author of the present article for the sake of argument. [VALLEX], p. 295, gives the lexical unit as reproduced in (1) without including the feature 'cor4'. According to the notational conventions used in FGP, this would ban the clitic reflexive *se* from appearing in constructions with the verb *vnímat* rendering (3) ungrammatical. Adding the feature 'cor4' to the entry in (1) would not, however, solve the problem as the valency frame requires an Effect argument marked for accusative case, independently of the presence or absence of a clitic reflexive. The frame would thus generate sentence (3) with *sílu* in the accusative case, which would be clearly ungrammatical in Czech. The feature 'cor4' is discussed further down in the main text of the article.

reflexive clitic *se* co-occurring with *vnímat* in (3) can be substituted with the strong form *sebe* without causing a change in lexical meaning. Contrary to what has been said above, the reflexive must be therefore classified as the clitic form of the reflexive pronoun. But if the reflexive co-occurring with *vnímat* in (3) is a reflexive pronoun, then the Corollary of Principle 2 states, that *vnímat* in (3) is to be regarded as belonging to the same lexical unit as *vnímat* in (2), where the same verb is co-occurring with a (non-reflexive) nominal.

From the discussion above we have to draw the conclusion that Principle 1 and Principle 2, applied to the verb *vnímat*, lead to an incoherent description. While Principle 1 (on the basis of the change in morphosyntactic form of one of the syntactic arguments) makes it necessary to treat the verb forms in (2) and (3) as homonymous surface forms of two different lexical units, Principle 2 (on the basis of the commutability of the reflexive, cf. (4)) requires the assignment of both verb forms to the same lexical unit.

Note that it is not feasible to simply assign every occurrence of *vnímat* paired with a clitic reflexive to one lexical unit ('vnímat se') while assigning all occurrences of *vnímat* without a clitic reflexive to another lexical unit ('vnímat'), presumably to the one given in (1). The reason is that 'vnímat' in (1) provides a valency slot for a Patient argument which is, as can be seen from (4), eligible for being filled by a reflexive pronoun. If *sebe* and *se* are two forms of the same paradigm differing solely in their prosodic weight, as it is claimed in FGP, there is no way to prevent the insertion of *se* in the Patient slot of 'vnímat' and, therefore, no way to categorically exclude the possibility of a surface form *vnímat se* being an instance of the syntagma 'vnímat' + 'Reflexive Pronoun'. In order to prevent the insertion of *se*, the syntax would have to be allowed to refer to the prosodic properties of certain expressions, which certainly is highly undesirable. (Cases of restriction to one prosodic form only as вести себя 'to behave' in Russian have probably to be treated as verbal idioms not formed by the ordinary rules of syntax, which is clearly not the case of *vnímat sebe* in Czech.) At least potentially, the clitic reflexive appearing in (3) is thus a possible instance of a reflexive pronoun and we are, therefore, obliged to apply the operational tests for identifying its type. In this case, we are still faced with the collision between Principle 1 and 2.

Before turning to possible solutions of the problem, it might be appropriate to take a look at a wider corpus of examples in order to determine how widespread the described collision is. This is done in the following section.

4. Further examples of collision

The examples presented in this section are taken from the electronic version[7] of [VALLEX] which allows for electronically scanning of the material. According to in-

[7] More precisely, I refer to version 2.6 of VALLEX accessible from `http://ufal.mff.cuni.cz/vallex/2.6/doc/home.html`. The research has been carried out in January, 2013.

formation on the homepage of [VALLEX], the electronic dictionary contains around 6,460 lexical units. Filtering the data with respect to functional roles reveals that 606 of these lexical units contain EFF in their valency frame. However, not all instances of EFF found in the dictionary are relevant to the present discussion.

First, there are numerous instances of EFF, which somehow depart from the prototypical case of a predicative complement in the sense of Sgall et al. (1986) as they designate some kind of spatial relation, possibly metaphorically extended to further kinds of relations. Examples taken from [VALLEX] are given in (5) and (6):

(5) smíchat mouku s vajíčky *v těsto*
'to mix flour and eggs *into a dough*'
Valency frame: ACT_1, $ADDR_{s+7}$, PAT_4, $EFF_{do+2,\ v+4}$

(6) Hájil se *proti naříčení*.
'He defended himself *against the slander*.'
Valency frame: ACT_1, PAT_4, $EFF_{proti+3,\ před+7}$

The underlying reason for classifying the arguments in (5) and (6) set in cursive as Effect seems to be an implicit small-clause analysis in the style of Generative Grammar (cf. Rothstein, 2009; Bailyn, 2012, among many others). For the purpose of the present article it is important to see that for principle reasons no collision between Principle 1 and Principle 2 can arise in these cases. The reason is that the Effect argument is syntactically to be realized as the complement of a preposition (*v* 'into' in (5), *proti* 'against' in (6)) which unambiguously determines its case value. No case alternation which would require setting up a separate valency frame can, therefore, arise. This can be clearly seen from example (6), where the substitution of the reflexive *se* for some other expression has no impact on the case marking on the Effect argument. The nominal *naříčení* 'slander' in *Hájil ho proti naříčení* 'He defended him against the slander' is still marked for dative case as required by the valency frame.

Neither a second type of valency frame containing an Effect argument is relevant to the present discussion. An example is given in (7):

(7) Pokládal ho *za přítele*.
'He took him *as a friend*.'
Valency frame: ACT_1, $PAT_{4,\ inf,\ že,\ cont}$, $EFF_{za+4,\ za+adj-4}$

Although a prototypical instance of a predicative complement as it expresses a genuine predicative relation ('he is a friend'), the Effect argument *za přítele* 'as a friend' in (7) shows the same formal behavior as the less prototypical cases discussed in the previous paragraph. It is embedded under the preposition *za* and thus invariably receives a fixed value for case. In this respect, the verbs *vnímat* in (2) and *pokládat* in (7), though nearly synonymous in the given contexts, differ in a crucial way. While the preposition *za* required by the valency frame in (7) assigns a stable case value to the Effect argument in (7), the element *jako* appearing with the Effect argument in (2) does

not.[8] This is, by the way, the reason why the Academic Grammar of Czech (Komárek et al., 1986, p. 225), treats *jako* not as a preposition, but as a conjunction. The direct consequence of this difference can be seen in comparing (2) and (3) above to (7) and (8) below.

(8) Pokládal se *za nejchytřejšího ze všech*.
'He regarded himself *as the most intelligent of all*.'

While the case value of the Effect argument in (2) and (3) alters in correlation to the presence or absence of the clitic reflexive, giving rise to the collision of principles diagnosed in Section 3, no such alternation can be seen in (7) and (8). The replacement of the non-reflexive pronoun *ho* 'him' in (7) for the clitic reflexive *se* in (8) has no repercussion on the case marking of the Effect argument (*za přítele* 'as a friend' in (7), *za nejchytřejšího ze všech* 'as the most intelligent of all' in (8)), which invariably appears in the accusative case. We have to conclude, then, that the only Effect arguments relevant to this study are those marked in the valency frame with the element *jako*.

Third, there are certain verbs selecting an Effect argument which for pragmatic reasons hardly ever appear in reflexive constructions of type (c), i.e. in constructions coding co-reference between two different arguments. The reasons for this incompatibility can be divided into several groups, the most important being certain selectional restrictions imposed on the participants and logical requirements on the number of participants implied in the verb meaning.

As for selectional restrictions, certain verbs do not allow for human participants as the Patient of the event they express. As the prototypical Actor of a transitive verb is human, no relation of co-reference can be established in this case because an ACT bearing the feature [+hum] cannot be referentially identical with a PAT bearing the feature [–hum]. A feature crash would be the inevitable result. A typical example is the verb *zařizovat/zařídit* 'to furnish' including PAT and EFF (marked with *jako*) in its valency frame. As one can furnish rooms and other spaces, cf. (9), but not human beings (including oneself), there can be no reflexive construction with the verb *zařizovat/zařídit* 'to furnish'.

(9) Zařídil si předsíň jako kuchyň.
'He furnished his hall as a kitchen.'

Further cases of blocked reflexivization include verbs coding situations in which the Actor and the Patient are necessarily different persons. For instance, the verb *to leave* lexically implies a separation of two entities from one another. As one and the same person cannot be separated from her/himself, the two roles of Actor and Patient have to be attributed to different participants and never to only one participant, as

[8] This cannot be seen from the valency frame in (1), where *jako* is associated with the accusative case. As I will argue further down in the article, this practice should be rejected.

1	*angažovat*	to engage
2	*brát*$_7$	to take
3	*deklarovat*$_2$	to declare
4	*fotografovat*$_1$	to take a picture
5	*hodnotit*$_1$	to evaluate
6	*chápat*$_2$	to perceive, to take as
7	*interpretovat*$_1$	to interpret
8	*nazývat/nazvat*$_1$	to call
9	*ohodnocovat/ohodnotit*$_1$	to rate
10	*označovat/označit*$_2$	to declare, to call
11	*pojímat/pojmout*$_3$	to comprehend, to conceive
12	*prezentovat*$_2$	to present
13	*udržovat/udržet*$_3$	to keep, to retain, to hold
14	*určovat/určit*$_3$	to appoint, to designate
15	*ustavovat/ustavit*$_2$	to establish
16	*usvědčovat/usvědčit*$_2$	to convict
17	*uznávat/uznat*$_2$	to acknowledge, to recognize
18	*vidět, vídat*$_5$	to see
19	*vnímat*$_3$	to perceive
20	*vyhlašovat/vyhlásit*$_2$	to proclaim
21	*znát*$_1$	to know

Table 1. Verbs from VALLEX leading to conflict between Principle 1 and 2

would be the case under reflexivization. An example is the verb *ponechávat* 'to leave behind, to leave alone'. In a sentence like *Ponechává děti samotné* 'He leaves the children alone', the word *děti* 'children' cannot be replaced by a reflexive without producing a nonsense statement. As a result, sentences of this type are rarely found in actual language data and are, therefore, irrelevant from a practical point of view.

In [VALLEX], the pragmatic restrictions on reflexivization lined out in the previous paragraphs are captured by the features 'cor4' and 'cor3' (cf. Žabokrtský and Lopatková, 2007, p. 53). The presence of these features in a lexical entry indicates, that the accusative position and/or the dative position are accessible for semantic reflexivization via the clitic reflexives *se* or *si*, respectively. However, it is not possible to use these features for automatic scanning of the material, as they are used somehow inconsistently. For example, the entry for *hodnotit* 'to evaluate' (p. 74) is equipped with the feature cor4, the entry for *vnímat* 'to regard as' (p. 295) is not. Example (3) above shows that semantic reflexivization of *vnímat* is, nevertheless, possible and actually quite usual. I therefore had to go through all entries for verbs requiring an Effect argument in order to decide on a case to case basis, if they are relevant to the

present study or not. The verbs I retained as relevant are listed in Table 1. (A few more comments on the feature 'cor4' will be shortly added in the next paragraph.)

In each single case, a conflict between Principle 1 and Principle 2 arises whenever the verb is used in a reflexive construction of the syntactic type (i.e. type c). Where [VALLEX] gives examples containing clitic reflexives, the conflict is apparent directly in the respective lexicon articles. In (10) and (11), I give examples taken from the electronic version of [VALLEX]. The parts of the entries incompatible with the actual language material (adduced as examples within the same dictionary article) are emphasized in bold-face.

(10) *pojímat/pojmout*$_3$
ACT$_1$, PAT$_4$, EFF**$_{\text{jako+4}}$**, jako+adj-4, za+4
Pojal se jako *nepostradatelný člen skupiny*.
'He regarded himself as *an indispensable member of the group*.'

(11) vidět, vídat$_5$ ACT$_1$, PAT$_4$, EFF$_7$, **jako+4**, Adj-4
Už se viděla jako jeho *nevěsta*.
'She already saw herself as his *bride*.'

As can be seen from (10) and (11), the nouns, resp. nominal groups, in the examples realizing the Effect argument are not in the case required by the corresponding valency frame. Instead of appearing in the accusative case, *nepostradatelný člen skupiny* 'an indispensable member of the group' and *nevěsta* 'bride' are in the nominative case. This violation of the co-occurrence restrictions stated in the respective valency frames remains without a comment in the dictionary.

Although the group of verbs showing the kind of syntactic misbehavior pointed out above is not numerous, the cases cannot be discarded as marginal either. Admittedly, the 21 units identified in Table 1 make up for merely 0.3% of the vocabulary covered in [VALLEX] (6,460 verbal lexical units). If one, however, takes into account only the units containing EFF in their valency frame (606 units), the percentage of relevant cases rises to 3.5%. In addition, verbs such as *nazývat/nazvat* 'to call' or *ohodnocovat/ohodnotit* 'to rate, to evaluate' have to be considered as highly frequent in modern Czech.[9] The search for a possibility to bring in line the analysis of reflexives and the principles for setting up lexical units is not a completely idle business, then.

5. Possible remedies for the conflict

In the present section, I will discuss two possibilities to resolve the conflict arising between Principle 1 and Principle 2. The first solution to the problem is to simply eliminate reflexivization of type (c) from the descriptive framework, thus undermining the basis of Principle 2. The second solution consists in relaxing the formal requirements

[9] Among 50,000 entries, the Frequency Dictionary of Czech (Čermák et al., 2004) assigns the frequency rank 2,014 to the verb *nazývat* (*se*) and the rank 9,196 to the verb *ohodnotit*.

on syntactic contexts to be counted as identical with respect to the valency frame of a lexical unit. In doing so, it will become possible to assign two verb forms which at present have to be considered (according to Principle 1) as to two different lexical units to the same unit.

Before turning to the concrete solutions, a further note on the features 'cor4' and 'cor3' seems appropriate. From the brief remarks in the previous sections it may seem that the whole problem of reflexivization in the context of an Effect argument boils down to occasional lexicographic mistakes in marking lexical units for these features. This is, however, not the case. By marking the presently deficient units such as *vnímat* 'to regard as' additionally for the feature 'cor4' one would only multiply the problem that poses the change in case value of the Effect argument for a coherent description. On the other hand, removing the feature 'cor4' from all units in question would not only fail to capture the data (as the respective verbs do occur in actual language data with the clitic reflexive *se*); it would be basically ineffective as the possibility of inserting a certain prosodic form of the reflexive pronoun cannot dependent on the presence or absence of a certain feature. As has been pointed out already in Section 3, the only precondition for inserting lexical items into the structure is the availability of the respective slot and possibly a match in subcategorization features such us [±animate]. If *se* and *sebe* are just two different prosodic variants of a certain form of one and the same lexical item not differing in grammatical features (both bearing the case value [accusative]), no rule of grammar should be able to distinguish them. A feature such as 'cor4' can therefore be nothing more than a notational device which makes it more convenient for non-native speakers of Czech to use the dictionary. In addition, it may facilitate the electronic parsing of language corpora by indicating how the homonymy of different occurrences of the clitic reflexive *se* is probabilistically to be resolved in a certain case at hand. It cannot be regarded however as some kind of subcategorization feature, as there is no linguistic basis for its application, at least not if one wishes to retain a unified paradigm for all personal reflexive pronouns.

5.1. Reflexivization in the lexicon

For Czech, the suggestion to eliminate reflexivization of type (c), i.e. the type of reflexivization where the reflexive clitic is considered to be a pronoun occupying a syntactic valency slot, goes back to Oliva (2000, 2001). Similar suggestions for the reflexive clitic *se* in French have been made earlier within the frameworks of Lexical Functional Grammar (Grimshaw, 1982) and Government and Binding Theory (Wehrli, 1986). According to these suggestions, the clitic *se* is viewed in general as a marker of valency reduction. Co-reference between two semantic arguments is, according to these suggestions, coded within the semantic structure of the derived lexical unit. Expanding on the account given by Oliva (2001, p. 204), the derivation of reflexive verbs from transitive non-reflexive verbs can be described as follows.

First, the semantic arguments implied in the semantic structure of the respective non-reflexive verb are lexically co-indexed, forcing a reflexive interpretation on the described event, (12).

Second, the surface valency of the derivational base is reduced by one slot. In consequence, only the higher semantic argument can be mapped onto a syntactic position, (13).

Finally, the clitic *se* is added to the structure, serving as visible sign that the operations described in (12) and (13) have taken place, (14).

(12) mýt (A-1, A-2) → mýt (A-1_i, A-2_i)

(13) mýt (ACT_1, PAT_4) → mýt (ACT_1)

(14) /mít/ → /mít se/

According to Oliva, the relevant syntactic properties of the derived reflexive verb such as the absence of agreement of secondary predicates with the reflexive in case features can now be deduced from the fact that the clitic is not a syntactic object at all. It therefore bears no case feature which could be spread within the syntactic structure. In the context of the present discussion, the account given by Oliva would solve the problem of having to treat surface forms such as *vnímat* and *vnímat se* as derived from the same lexical unit 'vnímat' (on the basis of the commutation test) and as derived from two different lexical units, i.e. 'vnímat' and 'vnímat se', (on the basis of the valency change EFF_4 → EFF_1) at the same time. As the clitic *se* is not a syntactic object, it could never be inserted into the PAT-slot of a valency frame in principle. There is, therefore, no longer a reason to worry about concomitant valency changes in the presence of *se*, as such changes can now be consistently attributed to the lexical operations (12)–(14) introducing *se* in the first place.

The general treatment of reflexivization as a lexical operation has, however, several undesirable consequences for the description of the Czech grammatical system as a whole. These consequences have been pointed out in the literature subsequent to the publication of Oliva (2001) and will not be repeated in detail here (cf. Komárek, 2001; Panevová, 2001). The most pertinent argument against a generalized lexical analysis of all clitic reflexives, it seems to me, is the fact that it neglects the obvious formal parallelism between the non-reflexive pronominal forms on the one hand (*tě – tebe*, *ti – tobě*) and the reflexive forms on the other (*se – sebe*, *si – sobě*), cf. Panevová (2001). There is, however, a further problem with the lexicalist approach, which never has been, as far as I know, explicitly pointed out in the literature, and which I wish to discuss in some greater depth here.

A convenient example to illustrate the problem is the verb *představit* 'to introduce', which implies three arguments in its valency frame: the one introducing someone (the Actor), the one introduced (the Patient) and the one to whom the patient is introduced (the Addressee). A possible context for the verb *představit* is given in (15):

(15) Předseda představil kandidáty členům poroty.
'The chair of the commission introduced the candidates to the members of the commission'

The verb *představit* is eligible to reflexivization of type (c), in which case a reciprocal reading arises, as shown in (16):

(16) Členové poroty si (navzájem) představili kandidáty.
'The members of the commission introduced the candidate to one another'

As (16) contains a reflexive clitic (*si*), a lexical derivation similar to the one described in (12)–(14) is called for. This hypothetical derivation is given in (17):

(17)	Lexical unit:	*představit*		*představit si*
	Semantic arguments:	A-1, A-2, A-3	→	A-1_i, A-2, A-3_i
	Valency slots:	ACT_1, $ADDR_3$, PAT_4		ACT_1, PAT_4

The reflexive *si* is of the dative kind, the operation, therefore, targets the third argument, i.e. the Addressee, which is co-indexed with the first argument, i.e. the Actor. The subsequent valency reduction affects the dative slot. The operation correctly captures the semantic interpretation of example (16), repeated in (18) together with the semantic roles assigned to the subject position. The members of the commission are both the Actors and the Addressees of the introducing event:

(18) Členové $\text{poroty}_{\text{A-1/A-3}}$ si (navzájem) představili kandidáty.

There is, however, a further possible, even though marginal, syntactic context for the realization of the lexical unit derived in (17). In principle, it could appear in a passive construction with the second argument (the Patient) promoted to the subject position:

(19) Kandidáti si byli představeni.
'The candidates where introduced to one another'

Let us see, if (17) provides the right interpretation for (19) as well. In a passive construction, the Patient of the derived entry in (17), i.e. A-2, which is not co-indexed with any further semantic argument, would be mapped onto the subject position. The nominal *kandidáti* in (19) thus receives the role of the one being introduced. The two remaining semantic arguments, bound together via lexical co-indexation, do not appear in the construction (19). Nevertheless, they have to be a part of the semantic interpretation, as they are fixed in the lexical entry in (17). According to the lexical entry, the argument not realized in (19) is the Actor and the Addressee of the introducing event at the same time. We thus derive an interpretation of (19), where the candidates are introduced by an unknown person or an unknown group of persons to this same person or group of persons.

(20) # Kandidáti$_{A\text{-}2}$ si byli představeni.
'The candidates are introduced (by someone to himself)'

As indicated by the sign '#', this interpretation (reflected by the English translation given in (20)), is not the correct interpretation of sentence (19). The correct interpretation, reflected by the English translation given in (19), cannot be deduced from the derived entry in (17), but has to be read off from the surface structure via co-indexation of the reflexive clitic with the subject position, assigning both the role of the Patient and the Addressee to the candidates: *kandidáti*$_{A\text{-}2/A\text{-}3}$.

(21) Kandidáti$_i$ si$_i$ byli představeni.

The syntax must obviously have access to the reflexive clitic *si* in order to provide the indices necessary for the right interpretation of (19). I conclude, then, that *si* must be a syntactic object eligible to syntactic co-indexation. From this it follows, that there must be instances of *se* being a syntactic object as well. The reason is that analyzing *si* as a syntactic object presupposes (at least under standard, weak-lexicalist assumptions) the existence of an abstract lexical unit (let us designate it by 'se'), which in dative contexts is realized by the word form *si*. Given the existence of such an abstract lexical unit, there is no way to ban the insertion of the same unit into accusative contexts, which presumably would cause its realization by the word form *se*. Accepting *si* as object clitic thus necessarily implies accepting *se* as object clitic as well,[10] and the argument denying object status to any occurrence of *se* can no longer be maintained.

There is, however, a possibility to guarantee the correct interpretation of (19) by lexical means alone. In order to do so, one would have to introduce a further lexical operation deriving a passive entry in a similar fashion as the reflexive entry in (12)–(14). The derived passive entry of the respective verb would then provide an altered argument structure (presumably with A-1/Actor removed from the accessible part of the frame) on which reflexivization could operate. Although strong lexicalist models of the sort obviously assumed by Oliva usually include operations of the described kind, I find this solution undesirable for several reasons:

- The generalization that passive constructions are lexically identical to active constructions, differing only in the way the semantic arguments are mapped onto syntactic positions, is lost.

[10] This statement holds within a weak lexicalist framework, unless one allows for defective paradigms of an extreme kind, i.e. for paradigms consisting solely of a dative form ('se': *si*). One could argue, that paradigms of this sort should be banned from description as ill-formed, as the defect affects a cell (the accusative cell) which is ranked higher in the commonly assumed case hierarchy (i.e. Nom > Acc > Dat > Gen) than a regularly filled cell (the dative cell). The absence of a nominative form for 'se', on the other hand, is no reason to reject the paradigm as ill-formed, as its absence can be attributed to syntactic rather than morphologic factors, i.e. the requirement for syntactic binding of the form, which in case of a nominative form (the subject) cannot be met.

- The distinction between fully productive inflectional processes and semi-productive derivational processes is blurred. In this context, one has to point out, that within FGP "passive" is regarded as one of the inflectional categories of the verb, cf. Panevová and Ševčíková (2014). This analysis could not be maintained when it is claimed that forming a passive is a derivational operation affecting a lexical unit targeted by the operation.
- There would have to be unmotivated, stipulated ordering between the operation of passivization (first) and reflexivization (subsequent to passivization) in order to prevent the derivation of passives from lexical entries such us (17).
- Whether a position inside the valency frame could be targeted by reflexivization or not, would have to be arbitrarily stipulated as well. There would be no principled reason that reflexivization could not target the Actor and the Addressee of a passive entry, generating deviant structures such us (21). In any case, the generalization that it is the *subject*, which serves us antecedent in a reflexive construction, would be lost.

Although the discussion above is not conclusive and does not exhaust the problem, it provides, I think, enough motivation for looking for alternative solutions to the lexicalist analysis. In the next section, I suggest an analysis in the spirit of Modified Valency Theory (Karlík, 2000a,b), which would allow us to retain *se* as reflexive clitic pronoun.

5.2. Formally underspecified valency frames and case attraction

Facing data such as (2)–(4) above, there seem to be three generalizations which should arguably be reflected in a grammatical description of Czech. First, the identity in meaning of the verb *vnímat* heading the three constructions and the commutability of the expressions *syna*, *se* and *sebe* (as long as we ignore the formal alternation in the concomitant *jako*-phrase) suggest, that the case should be handled in the syntax, not in the lexicon. This corresponds to a traditional architecture of grammar, according to which productive and semantically inert processes are to be located in the syntactic module, while idiosyncratic and semantically active processes belong into the lexicon (e.g., Karlík, 2000b). Second, the change in formal features of the Effect argument in (2) vs. (3) is obviously not due to a genuine change in valency but reflects the altered referential properties of the construction, namely the fact, that the Effect argument in (2) is co-referential with the Patient argument of the verb only (and shares, therefore, the accusative case), while in (3) it is in addition co-referential with the Actor argument (and shares, therefore, the nominative case). The case feature on the nominal within the *jako*-phrase should, therefore, be independent of the valency frame, as it is not lexically determined by the valency carrier, but by the syntactic context in which this valency carrier is realized. This observation is further confirmed by passive constructions such as (22) or (23), where the Patient argument appears in the subject position and is case-marked for nominative. As a result, the Effect argument,

contrary to what is stated in the respective valency frames given in [VALLEX], is case-marked for nominative as well. The generalization, then, is obviously that there is a dependency between two case values within the structure.

(22) Karel IV. byl vnímán jako třetí král.
'Charles was regarded as the third king.'

(23) Petr byl označen jako blbec.
'Peter has been called a fool.'

Third, as the contrast between (3) and (4) shows, co-reference alone is not sufficient for determining the case feature appearing on the Effect argument. There seems to be a notion of prominence in play, connected to Topic-Focus-Articulation (Sgall et al., 1986, p. 176) or Functional Sentence Perspective (Firbas, 1992). As soon as the reflexive element is emphasized and surfaces therefore as *sebe*, cf. (4), its own case value (accusative case) prevails over the competing case value of its antecedent. If, on the other hand, the reflexive element is deemphasized and surfaces, therefore, as reflexive clitic *se*, cf. (3), the case value of the antecedent (nominative case) takes precedence over the case value of the reflexive (accusative case).

To capture these generalizations, I make the following five suggestions.

Suggestion 1 The commutation test should be considered as valid in determining the syntactic status of the reflexive clitic at hand. A commutating reflexive clitic is to be analyzed as syntactic material realizing one of the arguments of the valency carrier and, therefore, as occupying a syntactic valency slot. The two occurrences of *vnímat* in (2) and (3) can thus be analyzed as lexically identical.

Suggestion 2 In order to avoid the resulting collision between Principle 1 and Principle 2, there must be drawn a distinction between structural case and lexical case. This distinction is well established at least since Chomsky, 1981 (cf. Woolford, 2006, for a more recent justification), and has been adapted for Valency Theory by Karlík (2000a). Following Karlík (2000a, p. 179), I consider a case appearing on a syntactic argument as "structurally assigned", if its value depends on the grammatical context in which the respective argument is realized. As structural cases take different values according to the grammatical context in which the respective argument appears, their value cannot be fixed in the lexical entry of the valency carrier (its valency frame) but has to be determined independently by general principles of grammar (for further details, see Karlík, 2000a, 2004). Although the prototypical cases analyzed as being "structural" are the nominative and the accusative case appearing on Actors and Patients of transitive constructions, the case appearing on the Effect argument is a clear instance of the same phenomenon. As has been demonstrated above, its value alter-

nates according to the syntactic context in which the Effect argument is realized, i.e., in accordance to the case value assigned to the Patient argument of the same valency carrier.

The theory of structural case suspends us from the obligation to determine the formal features of every single argument at the level of the valency frame. Arguments to be realized in structural valency slots can be notated in the valency frame without any explicit formal specification. As a consequence, it becomes possible to derive different surface constructions, such as the active and passive diathesis, among others, directly from the same valency frame. In the context of the present discussion, assuming structural cases renders it unnecessary to provide two different lexical entries for (2)/(4) and (3). As the Effect argument (being subject to structural case assignment) will no longer be specified in the valency frame as to its formal case feature, all three constructions can be derived from the same valency frame. The three occurrences of *vnímá* in (2)–(4) can, therefore, be analyzed as different realizations of *the same lexical unit*. By the same token, there is no longer any collision between Principle 1 and Principle 2. On the basis of the commutating reflexive, Principle 2 identifies *vnímá* (*někoho*) and *vnímá* (*se*) as instances of the same lexical unit, while Principle 1 can be satisfied by providing an identical valency frame for both occurrences of this unit.

Suggestion 3 In order to guarantee the correct value of the case feature appearing on the Effect argument, i.e. a value identical to the value of the case feature appearing on the Patient argument, some additional technical device has to be used. Obviously, some form of indexation is needed to copy the case value on the Patient argument onto the Effect argument. The linguistic reason behind this technical solution is the fact that the Effect argument is realized in a structural position to which in principle no case value can be assigned directly. This is so, because the position is embedded under the element *jako*, which does not assign case. The case value has, therefore, to be provided by other means than mere configuration within a structure, in the case at hand apparently by a referential dependency.

In the spirit of Modified Valency Theory but sticking to the notational convention of FGP, I suggest the following entry for the lexical unit 'vnímat' appearing in (2)–(4). The entry (24) is a revised version of the original entry (1) introduced in Section 3.

(24) Revised entry for $vnímat_3$
Meaning: 'to see, to regard, to consider, (as)'
Valency frame: ACT, PAT_i, $EFF_{jako+(i)}$.

When the verb 'vnímat' is realized inside a canonical transitive construction in the active diathesis, ACT and PAT receive (in accordance to general principles of valency realization) nominative and accusative case, respectively. Via co-indexation the case value "accusative" assigned to PAT then will be passed on to EFF. This is the state of affairs illustrated in (2) above. When the verb 'vnímat' is realized inside a passive

construction, PAT receives (in accordance to principles of non-canonical valency realization) nominative case which, again, will be passed on to EFF. This is the state of affairs illustrated in (22) above.

A further distribution of case values for structural case can be seen from the examples in (25) which have been suggested by an anonymous reviewer:

(25) a. Klaus vnímá své soky jako hráč.
'Klaus takes his rivals as a sportsman.'
b. Jak vnímáte Prahu jako architekt?
'What do you as an architect think of Prague?'

The nominals presumably realizing the Effect argument of the valency frame in (24), i.e. *hráč* 'player, sportsman' in (25a) and *architekt* 'architect' in (25b), are marked for nominative case. If the case value is passed on via co-indexation from the Patient argument to the Effect argument, as stated in (24), this should not be possible. The obvious reason for the appearance of the nominative case on these nominals is the fact that they are instances of subject-oriented depictive predicates (cf. Rothstein, 2009, p. 210) and as such are classifying the referent of the Actor, not the referent of the Patient. In order to capture this fact one would have to alter the co-indexation provided by the entry in (24) to an indexation connecting EFF to ACT.

There are several technical ways to achieve this. One could device a further lexical unit in which the altered co-indexation would be explicitly stated within the valency frame: ACT_i, PAT, $EFF_{jako+(i)}$. Considering the general applicability of the process to a wide range of lexical units, this solution would produce a huge redundancy in the lexicon and fail to draw the borderline between grammatical and lexical information. An alternative solution would be to remove all indices from the valency frame and to leave it to the syntax to provide the indices required for the intended semantic interpretation. One further could wonder if the depictives in (25) are really realizations of an Effect argument. As they provide only additional information on one of the arguments, they arguably have to be regarded as free modifications (or adjuncts, in generative terminology) and should therefore be excluded from the valency frame. The sentences in (25) could be derived from a lexical unit different from (24) which would not include an Effect argument in its valency frame.[11] The case marking on the depictive, then, would have to be handled independently of valency realization, probably along the lines of Panevová (1980, p. 79-86). This last solution seems feasible especially for (25b), less so for (25a), where the depictive is essential to the content of the sentence.

[11] [VALLEX], p. 295, provides one further lexical unit within the lexeme 'vnímat' (unit 4) which has the meaning 'to consider' and implicates only two arguments (Actor and Patient). The frame contains a position for a manner adverbial (MANN) which is to be considered as a free modification, not as an argument, cf. p. 20–21.

Whatever solution one chooses to adopt, it poses no additional problems for case assignment under reflexivization in (3) and (4), repeated below as (26) and (27), to which I turn in the following paragraph.

(26) Sám se vnímá jako „síla ochraňující divadlo“. (= 3)
'He sees himself as a force sheltering theatre'

(27) Karel IV. vnímá sebe jako vyvoleného třetího krále. (= 4)
'Charles IV sees himself as the chosen third king'

In the presence of a reflexive pronoun, the syntax provides (in addition to the lexically determined indices) further indices forcing a co-referential interpretation onto ACT and PAT. We end up, therefore, with an extended chain of indices: ACT_i, PAT_i, $EFF_{jako+(i)}$. As a result, the Effect argument now receives two different case values: nominative case via co-indexation with ACT (*pro* in 26, *Karel IV.* in 27) and accusative case via co-indexation with PAT (*se* in 26, *sebe* in 27). In order for the model to work, we must, thus, make further provisions to resolve this conflict between the two mutually exclusive case values.

Suggestion 4 A situation where two conflicting values for case are assigned to one and the same syntactic position is not uncommon in the languages of the world. Börjars and Vincent (2000) discuss relevant cases and suggest "resolution-rules" for several languages which predict the way the conflict is resolved. In our case, a resolution-rule giving precedence to the case value higher in the commonly assumed case hierarchy NOM > ACC > DAT > GEN (cf. Karlík, 2000b, p. 180; Caha, 2009, p. 291, for recent implementations of such a hierarchy) seems to be warranted. An EFF to which the values ⟨case: nominative, accusative⟩ have been assigned will, thus, be realized with the nominative case. This corresponds to the situation in (26), where the nominal *síla* 'force' is in the nominative case.

The solution to the problem of determining the appropriate case value for an Effect argument suggested above, i.e. using indexation to transfer case values onto EFF and applying subsequently a resolution-rule to favor a nominative value over a competing accusative value, can be regarded as a more formal restatement of a proposal made earlier by Panevová (2001). Panevová suggests treating the nominative case appearing on co-predicates in reflexive constructions as the result of "case attraction", where the expected accusative case (due to the non-distinctiveness of case in the given position) is overridden by a presumably more unmarked nominative case. If we accept this kind of reasoning, we now have to explain why the same resolution-rule does not lead to the same result in (27), where the Effect argument is marked with accusative case.

Suggestion 5 For explaining the accusative case appearing on EFF in (27), we have to take into consideration the Topic-Focus-Articulation of the example. Obviously,

the reflexive pronoun is under focus in (27) and thus surfaces in its emphatic mutation *sebe*. To capture the correlation between the case marking on EFF and the Topic-Focus-Articulation of the respective sentence, I suggest a formal pairing of the case value assigned to a nominal with a feature reflecting the position of the nominal within the Topic-Focus-Articulation of the sentence. This may seem problematic in a formal framework given the somehow vague status of properties related to the informational organization of the sentence;[12] it is, however, completely unproblematic within the framework of FGP, as the Topic-Focus-Articulation is treated in FGP as a grammatical property of the tectogrammatical structure of the sentence formally represented by grammatical features.[13] We can thus further specify the accusative case value assigned to the reflexive *sebe* in (27) with the formal feature "f" (for "focus"). Via co-indexation with *sebe* (PAT), the Effect argument receives, then, the following set of case values: ⟨case: nominative, accusative$_f$⟩. If we now formulate the following resolution-rule for conflicting case assignment, we correctly derive both (26) and (27):

(28) *Resolution-Rule for Czech*:
If different case values are assigned to one and the same position, only the hierarchically higher case value is morphologically realized, unless a lower value is paired with a focus-feature. In the latter case, the case value paired with the focus-feature is morphologically realized.

6. Some preliminary conclusions

In the present article, I have been discussing two different strategies to amend the collision between the principle of lexical identity (Principle 1) and the principle of pronominal commutability (Principle 2) which arises when these principles are applied to constructions of type (2) and (3)/(26). An apparently simple solution can be achieved in implementing a radical approach to reflexivization, treating every single instance of the reflexive clitic *se* as the output of a derivational operation on lexical entries. The price one has to pay is to deny the lexical identity of verb forms which (according to meaning) should arguably be treated as lexically identical. In addition, one has to derive passive structures in the lexicon as well, as the derivation of lexical entries for passive verbs must precede the derivation of lexical entries for reflexive verbs, as has been shown above in connection with dative-type reflexives. An alternative strategy to resolve the conflict between both principles makes it necessary to

[12] More recently, there is a certain change in attitude within the generative literature concerning questions of the informational organization of sentences. Especially in work developing the "cartographic" approach to sentence structure (cf. Cinque and Rizzi, 2008), topic phrases and focus phrases are treated on a pair with other phrasal projections.

[13] In the Prague Dependency Treebank (accessible from `http://ufal.mff.cuni.cz/pdt3.0/`), the nodes of a dependency tree are equipped with the attribute "tfa" (i.e. Topic-Focus-Articulation), which can bear three different values: t (for "topic"), f (for "focus") or c (for "contrastive").

allow for valency frames which are underspecified with respect to the formal features of certain arguments. Admittedly, one further needs quite complicate analytical machinery, including case transfer via co-indexation and a rule such as (28) resolving the conflict arising from double case assignment. Last but not least, one has to accept a focus-feature as part of the formal vocabulary of the grammar.

At this point, a final evaluation of both strategies seems to be in place, from which, however, I will refrain, as a serious evaluation would require a broader discussion of the grammatical framework implicitly presupposed by both solutions. That it is not possible to rely on mere simplicity of description with reference to an isolated detail of language structure, has been demonstrated in detail already by Matthews (1972). Instead, I will try to relate both solutions to the framework of FGP in its present state of development. In this respect, the suggestion made by Oliva (2001) seems to require some major changes in the framework of FGP. It would lead not only to the disintegration of the nominal paradigm (containing in the current state "weak" and "strong" forms) but also to the loss of the passive as a grammatical category of the Czech verb. In the final end it might undermine the distinction between inflection and derivation inherited by FGP from traditional grammar. In comparison to these fundamental changes, the adjustments needed when adopting the second solution seem to me of relatively minor importance. Basically, they come down to allowing valency frames underspecified for case values of arguments to be realized with structurally assigned case. Some mechanism of case transfer is required in Czech anyway in order to handle case marking on co-predicates not subcategorized by the governing verb, cf. Panevová (1980, p. 79-86). Something similar can be said with respect to the required focus feature, which is available within the framework independently of the proposed adjustment. If there is a need elsewhere in Czech grammar to apply a resolution-rule such as (28), which would obviously foster its status, remains to be seen. Which strategy one prefers in solving the collision between both principles may depend on one's personal preferences; it depends, however, just as much on the analytical possibilities of one's theoretical apparatus.

Bibliography

Ágel, Vilmos and Klaus Fischer. *Dependency grammar and valency theory*, pages 223–255. Oxford University Press, 2010.

Bailyn, John Frederick. *The Syntax of Russian*. Cambridge University Press, Cambridge, UK, 2012.

Börjars, Kersti and Nigel Vincent. Multiple case and the ‚wimpiness' of morphology. In Butt, Miriam and Tracy Holloway Kink, editors, *Argument Realization (Studies in Constraint-Based Lexicalism)*, pages 15–40, Stanford, CA, 2000. CSLI Publ.

Caha, Pavel. Poznámky k syntaxi předložky před. *Slovo a slovesnost*, 70(4):287–294, 2009.

Čermák, František, Michal Křen, et al. *Frekvenční slovník češtiny*. Nakladatelství Lidové noviny, Praha, 2004.

Chomsky, Noam. *Lectures on Government and Binding*. Mouton de Gruyter, Berlin and New York, 1981. Cited from the Seventh Edition (1993).

Cinque, Guglielmo and Luigi Rizzi. The cartography of syntactic structures. In Moscati, Vincenzo, editor, *CISCL Working Papers on Language and Cognition*, pages 43–59, 2008.

Daneš, František, Zdeněk Hlavsa, et al. *Větné vzorce v češtině*. Academia, Praha, 1987.

Firbas, Jan. *Functional sentence perspective in written and spoken communication*. Cambridge University Press, Cambridge, 1992.

Grimshaw, Jane. On the lexical representation of Romance reflexive clitics. In Bresnan, Joan, editor, *The mental representation of grammatical relations*, pages 87–148, Cambridge, MA, 1982. The MIT Press.

Karlík, Petr. Hypotéza modifikované valenční teorie. *Slovo a slovesnost*, 61(3):170–189, 2000a.

Karlík, Petr. Valence substantiv v modifikované valenční teorii. In *Čeština – univerzália a specifika 2. Sborník konference ve Šlapanicích u Brna*, pages 181–192, Brno, Czech Republic, 2000b. Masarykova univerzita.

Karlík, Petr. Pasivum v češtině. *Slovo a slovesnost*, 65(2):83–112, 2004.

Komárek, Miroslav. Několik poznámek k reflexi reflexivity reflexiv. *Slovo a slovesnost*, 62(3): 207–209, 2001.

Komárek, Miroslav, Jan Kořenský, and Jan Petr. *Mluvnice češtiny 2. Tvarosloví*. Academia, Praha, 1986.

Lopatková, Markéta, Zdeněk Žabokrtský, and Václava Kettnerová. *Valenční slovník českých sloves*. Nakladatelství Karolinum, Praha, 2008. ISBN 978-80-246-1467-0.

Matthews, Peter H. *Inflectional morphology. A theoretical study based on aspects of Latin verb conjugation*. Cambridge University Press, 1972.

Oliva, Karel. Hovory k sobě/si/sebe/se. In Hladká, Zdeňka and Petr Karlík, editors, *Čeština – univerzália a specifika*, pages 167–171, Brno, 2000. Masarykova univerzita.

Oliva, Karel. Reflexe reflexivity reflexiv. *Slovo a slovesnost*, 62(3):200–207, 2001.

Panevová, Jarmila. *Formy a funkce ve stavbě české věty*. Academia, Praha, 1980.

Panevová, Jarmila. Ještě k teorii valence. *Slovo a slovesnost*, 59(1):1–14, 1998.

Panevová, Jarmila. Česká reciproční zájmena a slovesná valence. *Slovo a slovesnost*, 60(4):269–275, 1999.

Panevová, Jarmila. Problémy reflexivního zájmena v češtině. In Kuklík, Jan and Jiří Hasil, editors, *Přednášky z XLIV. běhu Letní školy slovanských studií*, pages 81–88, Prague, 2001. UK v Praze, FF. ISBN 80-7308-004-4.

Panevová, Jarmila. Problémy se slovanským reflexivem. *Slavia*, 77(1-3):153–163, 2008. ISSN 0037-6736.

Panevová, Jarmila and Magda Ševčíková. Delimitation of information between grammatical rules and lexicon. (in print), 2014.

Rothstein, Susan. Secondary predication. In Everaert, Martin and Henk van Riemsdijk, editors, *The Blackwell Companion to Syntax*, volume IV., pages 209–233. Blackwell Publ., Oxford, 2009.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands, 1986.

SYN2005. Czech National Corpus. Synchronic Corpus of Contemporary Written Czech. URL `http://ucnk.ff.cuni.cz/english/syn2005.php`.

Wehrli, Eric. On some properties of French clitic se. In Borer, Hagit, editor, *Syntax and Semantics vol. 19: The syntax of pronominal clitics*, pages 263–283. Academic Press, Orlando, 1986.

Woolford, Ellen. Lexical case, inherent case, and argument structure. *Linguistic Inquiry*, 37(1): 111–130, 2006.

Wunderlich, Dieter. Cause and the structure of verbs. *Linguistic Inquiry*, 28(1), 1997.

Žabokrtský, Zdeněk and Markéta Lopatková. Valency information in VALLEX 2.0: Logical structure of the lexicon. *The Prague Bulletin of Mathematical Linguistics*, 87:41–60, 2007. ISSN 0032-6585.

Address for correspondence:
Roland Wagner
`wagner@ped.muni.cz`
Katedra německého jazyka a literatury
Pedagogická fakulta Masarykovy univerzity
Poříčí 7, 603 00 Brno
Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 101 APRIL 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 two copies 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`.