Influence of Parser Choice on Dependency-Based MT

Martin Popel, David Mareček, Nathan Green and Zdeněk Žabokrtský
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{popel,marecek,green,zabokrtsky}@ufal.mff.cuni.cz

Abstract

Accuracy of dependency parsers is one of the key factors limiting the quality of dependency-based machine translation. This paper deals with the influence of various dependency parsing approaches (and also different training data size) on the overall performance of an English-to-Czech dependency-based statistical translation system implemented in the Treex framework. We also study the relationship between parsing accuracy in terms of unlabeled attachment score and machine translation quality in terms of BLEU.

1 Introduction

In the last years, statistical n-gram models dominated the field of Machine Translation (MT). However, their results are still far from perfect. Therefore we believe it makes sense to investigate alternative statistical approaches. This paper is focused on an analysis-transfer-synthesis translation system called TectoMT whose transfer representation has a shape of a deep-syntactic dependency tree. The system has been introduced by Žabokrtský et al. (2008). The translation direction under consideration is English-to-Czech.

It has been shown by Popel (2009) that the current accuracy of the dependency parser employed in this translation system is one of the limiting factors from the viewpoint of its output quality. In other words, the parsing phase is responsible for a large portion of translation errors. The biggest source of translation errors in the referred study was (and probably still is) the transfer phase, however the proportion has changed since and the relative importance of the parsing phase has grown, because the tranfer phase errors have already been addressed by improvements based on Hidden Markov Tree Models for lexical and syntactic choice as shown by Žabokrtský and Popel (2009), and by context sensitive translation models based on maximum entropy as described by Mareček et al. (2010).

Our study proceeds along two directions. First, we train two state-of-the-art dependency parsers on training sets with varying size. Second, we use five parsers based on different parsing techniques. In both cases we document the relation between parsing accuracy (in terms of Unlabeled Attachment Score, UAS) and translation quality (estimated by the well known BLEU metric).

The motivation behind the first set of experiments is that we can extrapolate the learning curve and try to predict how new advances in dependency parsing can affect MT quality in the future.

The second experiment series is motivated by the hypothesis that parsers based on different approaches are likely to have a different distribution of errors, even if they can have competitive performance in parsing accuracy. In dependency parsing metrics, all types of incorrect edges typically have the same weight,[1] but some incorrect edges can be more harmful than others from the MT viewpoint. For instance, an incorrect attachment of an adverbial node is usually harmless, while incorrect attachment of a subject node might have several negative conse-

[1] This issue has been tackled already in the parsing literature; for example, some authors disregard placement of punctuation nodes within trees in the evaluation (Zeman, 2004).

Proceedings of the 6th Workshop on Statistical Machine Translation, pages 433–439,
Edinburgh, Scotland, UK, July 30–31, 2011.

quences such as:

- unrecognized finiteness of the governing verb, which can lead to a wrong syntactization on the target side (an infinitive verb phrase instead of a finite clause),
- wrong choice of the target-side verb form (because of unrecognized subject-predicate agreement),
- missing punctuation (because of wrongly recognized finite clause boundaries),
- wrong placement of clitics (because of wrongly recognized finite clause boundaries),
- wrong form of pronouns (personal and possessive pronouns referring to the clause's subject should have reflexive forms in Czech).

Thus it is obvious that the parser choice is important and that it might not be enough to choose a parser, for machine translation, only according to its UAS.

Due to growing popularity of dependency syntax in the last years, there are a number of dependency parsers available. The present paper deals with five parsers evaluated within the translation framework: three genuine dependency parsers, namely the parsers described in (McDonald et al., 2005), (Nivre et al., 2007), and (Zhang and Nivre, 2011), and two constituency parsers (Charniak and Johnson, 2005) and (Klein and Manning, 2003), whose outputs were converted to dependency structures by Penn Converter (Johansson and Nugues, 2007).

As for the related literature, there is no published study measuring the influence of dependency parsers on dependency-based MT to our knowledge.[2]

The remainder of this paper is structured as follows. The overall translation pipeline, within which the parsers are tested, is described in Section 2. Section 3 lists the parsers under consideration and their main features. Section 4 summarizes the influence of the selected parsers on the MT quality in terms of BLEU. Section 5 concludes.

[2]However, the parser bottleneck of the dependency-based MT approach was observed also by other researchers (Robert Moore, personal communication).

2 Dependency-based Translation in Treex

We have implemented our experiments in the Treex software framework (formerly TectoMT, introduced by Žabokrtský et al. (2008)), which already offers tool chains for analysis and synthesis of Czech and English sentences.

We use the tectogrammatical (deep-syntactic) layer of language representation as the transfer layer in the presented MT experiments. Tectogrammatics was introduced by Sgall (1967) and further elaborated within the Prague Dependency Treebank project (Hajič et al., 2006). On this layer, each sentence is represented as a tectogrammatical tree, whose main properties (from the MT viewpoint) are the following:

1. nodes represent autosemantic words,
2. edges represent semantic dependencies (a node is an argument or a modifier of its parent),
3. there are no functional words (prepositions, auxiliary words) in the tree, and the autosemantic words appear only in their base forms (lemmas). Morphologically indispensable categories (such as number with nouns or tense with verbs, but not number with verbs as it is only imposed by agreement) are stored in separate node attributes (grammatemes).

The intuitions behind the decision to use tectogrammatics for MT are the following: we believe that (1) tectogrammatics largely abstracts from language-specific means (inflection, agglutination, functional words etc.) of expressing non-lexical meanings and thus tectogrammatical trees are supposed to be highly similar across languages, (2) it enables a natural transfer factorization,[3] (3) and local tree contexts in tectogrammatical trees carry more information (especially for lexical choice) than local linear contexts in the original sentences.

The translation scenario is outlined in the rest of this section.

[3]Morphological categories can be translated almost independently from lemmas, which makes parallel training data 'denser', especially when translating from/to a language with rich inflection such as Czech.

2.1 Analysis

The input English text is segmented into sentences and tokens. The tokens are lemmatized and tagged with Penn Treebank tags using the Morce tagger (Spoustová et al., 2007). Then one of the studied dependency parsers is applied and a surface-syntax dependency tree (analytical tree in the PDT terminology) is created for each sentence.

This tree is converted to a tectogrammatical tree. Each autosemantic word with its associated functional words is collapsed into a single tectogrammatical node, labeled with a lemma, formeme,[4] and semantically indispensable morphologically categories; coreference is also resolved.

2.2 Transfer

The transfer phase follows, whose most difficult part consists especially in labeling the tree with target-side lemmas and formemes. There are also other types of changes, such as node addition and deletion. However, as shown by Popel (2009), changes of tree topology are required relatively infrequently due to the language abstractions on the tectogrammatical layer.

Currently, translation models based on Maximum Entropy classifiers are used both for lemmas and formemes (Mareček et al., 2010). Tree labeling is optimized using Hidden Tree Markov Models (Žabokrtský and Popel, 2009), which makes use of target-language dependency tree probabilistic model.

All models used in the transfer phase are trained using training sections of the Czech-English parallel corpus CzEng 0.9 (Bojar and Žabokrtský, 2009).

2.3 Synthesis

Finally, surface sentence shape is synthesized from the tectogrammatical tree, which is basically the reverse operation of the tectogrammatical analysis. It consists of adding punctuation and functional words, spreading morphological categories according to grammatical agreement, performing inflection (using Czech morphology database (Hajič, 2004)), arranging word order etc.

The difference from the analysis phase is that there is not very much space for optimization in the synthesis phase. In other words, final sentence shape is determined almost uniquely by the tectogrammatical tree (enriched with formemes) resulting from the transfer phase. However, if there are not enough constraints for a unique choice of a surface form of a lemma, then a unigram language model is used for the final decision. The model was trained using 500 million words from the Czech National Corpus.[5]

3 Involved Parsers

We performed experiments with parsers from three families: graph-based parsers, transition-based parsers, and phrase-structure parsers (with constituency-to-dependency postprocessing).

3.1 Graph-based Parser

In graph-based parsing, we learn a model for scoring graph edges, and we search for the highest-scoring tree composed of the graph's edges. We used Maximum Spanning Tree parser (Mcdonald and Pereira, 2006) which is capable of incorporating second order features (`MST` for short).

3.2 Transition-based Parsers

Transition-based parsers utilize the shift-reduce algorithm. Input words are put into a queue and consumed by shift-reduce actions, while the output parser is gradually built. Unlike graph-based parsers, transition-based parsers have linear time complexity and allow straightforward application of non-local features.

We included two transition-based parsers into our experiments:

- `Malt` – Malt parser introduced by Nivre et al. (2007) [6]

[4] Formeme captures the morphosyntactic means which are used for expressing the tectogrammatical node in the surface sentence shape. Examples of formeme values: `v:that+fin` – finite verb in a subordinated clause introduced with conjunction *that*, `n:sb` – semantic noun in a subject position, `n:for+X` – semantic noun in a prepositional group introduced with preposition *for*, `adj:attr` – semantic adjective in an attributive position.

[5] http://ucnk.ff.cuni.cz

[6] We used *stackeager* algorithm, *liblinear* learner, and the enriched feature set for English (the same configuration as in pretrained English models downloadable at http://maltparser.org.

- `ZPar` – Zpar parser[7] which is basically an alternative implementation of the Malt parser, employing a richer set of non-local features as described by Zhang and Nivre (2011).

3.3 CFG-based Tree Parsers

Another option how to obtain dependency trees is to apply a constituency parser, recognize heads in the resulting phrase structures and apply a recursive algorithm for converting phrase-structure trees into constituency trees (the convertibility of the two types of syntactic structures was studied already by Gaifman (1965)).

We used two constituency parsers:

- `Stanford` – The Stanford parser (Klein and Manning, 2003),[8]

- `CJ` – a MaxEnt-based parser combined with discriminative reranking (Charniak and Johnson, 2005).[9]

Before applying the parsers on the text, the system removes all spaces within tokens. For instance U. S. becomes U.S. to restrict the parsers from creating two new tokens. Tokenization built into both parsers is bypassed and the default tokenization in Treex is used.

After parsing, Penn Converter introduced by Johansson and Nugues (2007) is applied, with the `-conll2007` option, to change the constituent structure output, of the two parsers, into CoNLL dependency structure. This allows us to keep the formats consistent with the output of both MST and MaltParser within the Treex framework.

There is an implemented procedure for creating tectogrammatical trees from the English phrase structure trees described by Kučerová and Žabokrtský (2002). Using the procedure is more straightforward, as it does not go through the CoNLL-style trees; English CoNLL-style trees differ slightly from the PDT conventions (e.g. in attaching auxiliary verbs) and thus needs additional postprocessing for our purposes. However, we decided to stick to Penn Converter, so that the similarity of the translation scenarios is maximized for all parsers.

3.4 Common Preprocessing: Shallow Sentence Chunking

According to our experience, many dependency parsers have troubles with analyzing sentences that contain parenthesed or quoted phrases, especially if they are long.

We use the assumption that in most cases the content of parentheses or quotes should correspond to a connected subgraph (subtree) of the syntactic tree. We implemented a very shallow sentence chunker (`SentChunk`) which recognizes parenthesed word sequences. These sequences can be passed to a parser first, and be parsed independently of the rest of the sentence. This was shown to improve not only parsing accuracy of the parenthesed word sequence (which is forced to remain in one subtree), but also the rest of the sentence.[10]

In our experiments, `SentChunk` is used only in combination with the three genuine dependency parsers.

4 Experiments and Evaluation

4.1 Data for Parsers' Training and Evaluation

The dependency trees needed for training the parsers and evaluating their UAS were created from the Penn Treebank data (enriched first with internal noun phrase structure applied via scripts provided by Vadas and Curran (2007)) by Penn Converter (Johansson and Nugues, 2007) with the `-conll2007` option (`PennConv` for short).

All the parsers were evaluated on the same data – section 23.

All the parsers were trained on sections 02–21, except for the Stanford parser which was trained on sections 01–21. We were able to retrain the parser models only for `MST` and `Malt`. For the other parsers we used pretrained models available on the Internet: `CJ`'s default model `ec50spfinal`, `Stanford`'s `wsjPCFG.ser.gz` model, and

[7] http://sourceforge.net/projects/zpar/ (version 0.4)

[8] Only the constituent, phrase based, parsed output is used in these experiments.

[9] We are using the default settings from the August 2006 version of the software.

[10] Edge length is a common feature in dependency parsers, so "deleting" parenthesed words may give higher scores to correct dependency links that happened to span over the parentheses.

ZPar's english.tar.gz. The model of ZPar is trained on data converted to dependencies using Penn2Malt tool,[11] which selects the last member of a coordination as the head. To be able to compare ZPar's output with the other parsers, we postprocessed it by a simple ConjAsHead code that converts this style of coordinations to the one used in CoNLL2007, where the conjuction is the head.

4.2 Reference Translations Used for Evaluation

Translation experiments were evaluated using reference translations from the new-dev2009 data set, provided by the organizors of shared translation task with the Workshop on Statistical Machine Translation.

4.3 Influence of Parser Training Data Size

We trained a sequence of parser models for MST and Malt, using a roughly exponentially growing sequence of Penn Treebank subsets. The subsets are contiguous and start from the beginning of section 02. The results are collected in Tables 1 and 2.[12]

#tokens	UAS	BLEU	NIST
100	0.362	0.0579	3.6375
300	0.509	0.0859	4.3853
1000	0.591	0.0995	4.6548
3000	0.623	0.1054	4.7972
10000	0.680	0.1130	4.9695
30000	0.719	0.1215	5.0705
100000	0.749	0.1232	5.1193
300000	0.776	0.1257	5.1571
990180	0.793	0.1280	5.1915

Table 1: The effect of training data size on parsing accuracy and on translation performance with MST.

The trend of the relation between the training data size and BLEU is visible also in Figure 1. It is obvious that increasing the training data has a positive effect on the translation quality. However, the pace of growth of BLEU is sublogarithmic, and becomes unconvincing above 100,000 training tokens. It indicates that given one of the two parsers integrated into our translation framework, increasing the parser training data alone would probably not lead to a substantial improvement of the translation performance.

[11] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

[12] To our knowledge, the best system participating in the shared task reaches BLEU 17.8 for this translation direction.

#tokens	UAS	BLEU	NIST
100	0.454	0.0763	4.0555
300	0.518	0.0932	4.4698
1000	0.591	0.1042	4.6769
3000	0.616	0.1068	4.7472
10000	0.665	0.1140	4.9100
30000	0.695	0.1176	4.9744
100000	0.723	0.1226	5.0504
300000	0.740	0.1238	5.1005
990180	0.759	0.1253	5.1296

Table 2: The effect of training data size on parsing accuracy and on translation performance with Malt.

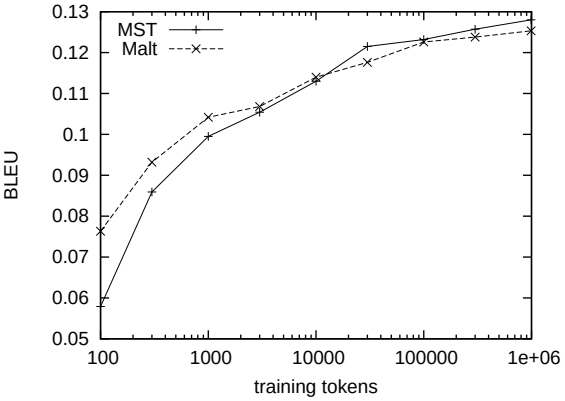

Figure 1: The effect of parser training data size of BLEU with Malt and MST parsers.

4.4 Influence of Parser Choice

Table 3 summarizes our experiments with the five parsers integrated into the tectogrammatical translation pipeline. Two configurations (with and without SentChunk) are listed for the genuine dependency parsers. The relationship between UAS and BLEU for (the best configurations of) all five parsers is depicted also in Figure 2.

Additionally, we used paired bootstrap 95% confidence interval testing (Zhang et al., 2004), to check which BLEU differences are significant. For the five compared parser (with SentChunk if applicable), only four comparisons are not significant: MST-CJ, MST-Stanford, Malt-Stanford, and CJ-Stanford.

Parser	Training data	Preprocessing	Postprocessing	UAS	BLEU	NIST	TER
MST	PennTB + PennConv	SentChunk	–	0.793	0.1280	5.192	0.735
MST	PennTB + PennConv	–	–	0.794	0.1236	5.149	0.739
Malt	PennTB + PennConv	SentChunk	–	0.760	0.1253	5.130	0.740
Malt	PennTB + PennConv	–	–	0.761	0.1214	5.088	0.744
Zpar	PennTB + Penn2Malt	SentChunk	ConjAsHead	0.793	0.1176	5.039	0.749
Zpar	PennTB + Penn2Malt	–	ConjAsHead	0.792	0.1127	4.984	0.754
CJ	PennTB	–	PennConv	0.904	0.1284	5.189	0.737
Stanford	PennTB	–	PennConv	0.825	0.1277	5.137	0.740

Table 3: Dependency parsers tested in the translation pipeline.

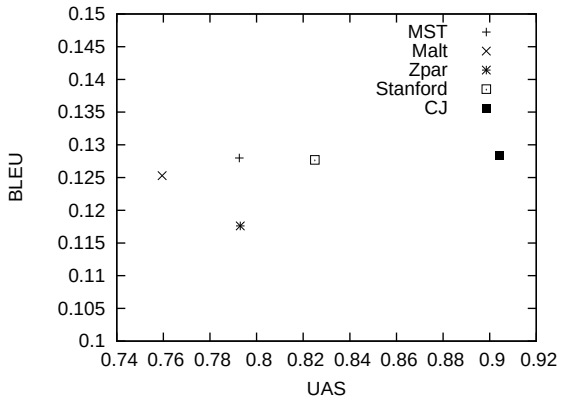

Figure 2: Unlabeled Attachment Score versus BLEU.

Even if BLEU grows relatively smoothly with UAS for different parsing models of the same parser, one can see that there is no obvious relation between UAS and BLEU accross all parsers. MST and Zpar have the same UAS but quite different BLEU, whereas MST and CJ have very similar BLEU but distant UAS. It confirms the original hypothesis that it is not only the overall UAS, but also the parser-specific distribution of errors what matters.

4.5 Influence of Shallow Sentence Chunking

Table 3 confirms that parsing the contents parentheses separately from the rest of the sentence (SentChunk) has a positive effect with all three dependency parsers. Surprisingly, even if the effect on UAS is negligible, the improvement is almost half of BLEU point which is significant for all the three parsers.

4.6 Discussion on Result Comparability

We tried to isolate the effects of the properties of selected parsers, however, the separation from other influencing factors is not perfect due to several technical issues:

- So far, we were not able to retrain the models for all parsers ourselves and therefore their pre-trained models (one of them based on slightly different Penn Treebank division) must have been used.
- Some parsers make their own choice of POS tags within the parsed sentences, while other parsers require the sentences to be tagged already on their input.
- The trees in the CzEng 0.9 parallel treebank were created using MST. CzEng 0.9 was used for training translation models used in the transfer phase of the translation scenario; thus these translation models might compensate for some MST's errors, which might handicap other parsers. So far we were not able to reparse 8 million sentence pairs in CzEng 0.9 by all studied parsers.

5 Conclusions

This paper is a study of how the choice of a dependency parsing technique influences the quality of English-Czech dependency-based translation. Our main observations are the following. First, BLEU grows with the increasing amount of training dependency trees, but only in a sublogarithmic pace. Second, what seems to be quite effective for translation

is to facilitate the parsers' task by dividing the sentences into smaller chunks using parenthesis boundaries. Third, if the parsers are based on different approaches, their UAS does not correlate well with their effect on the translation quality.

Acknowledgments

This research was supported by the grants MSM0021620838, GAUK 116310, GA201/09/H057, and by the European Commission's 7th Framework Program (FP7) under grant agreements n° 238405 (CLARA), n° 247762 (FAUST), and n° 231720 (EuroMatrix Plus).

References

Ondřej Bojar and Zdeněk Žabokrtský. 2009. CzEng 0.9, Building a Large Czech-English Automatic Parallel Treebank. *The Prague Bulletin of Mathematical Linguistics*, 92:63–83.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, pages 304–337.

Jan Hajič et al. 2006. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia.

Jan Hajič. 2004. *Disambiguation of Rich Inflection – Computational Morphology of Czech*. Charles University – The Karolinum Press, Prague.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics*, pages 423–430.

Ivona Kučerová and Zdeněk Žabokrtský. 2002. Transforming Penn Treebank Phrase Trees into (Praguian) Tectogrammatical Dependency Trees. *The Prague Bulletin of Mathematical Linguistics*, (78):77–94.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. 2010. Maximum entropy translation model in dependency-based MT framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 201–201, Uppsala, Sweden. Association for Computational Linguistics.

Ryan Mcdonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT / EMNLP*, pages 523–530, Vancouver, Canada.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Martin Popel. 2009. Ways to Improve the Quality of English-Czech Machine Translation. Master's thesis, Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic.

Petr Sgall. 1967. *Generativní popis jazyka a česká deklinace*. Academia, Prague.

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

David Vadas and James Curran. 2007. Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic, June. Association for Computational Linguistics.

Zdeněk Žabokrtský and Martin Popel. 2009. Hidden Markov Tree Model in Dependency-based Machine Translation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 145–148, Suntec, Singapore.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. 2008. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL*, pages 167–170.

Daniel Zeman. 2004. *Parsing with a Statistical Dependency Model*. Ph.D. thesis, Faculty of Mathematics and Physics, Charles University in Prague.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *To appear in the Proceedings of the 49th Annual Meeting of the Association of Computational Linguistics*.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system. In *Proceedings of LREC*, volume 4, pages 2051–2054.