Improvements to Dependency Parsing Using Automatic Simplification of Data

Tomáš Jelínek
Charles University
Prague, Czech Republic
Email: tomas.jelinek@ff.cuni.cz

Abstract

In dependency parsing, much effort is devoted to the development of new methods of language modeling and better feature settings. Less attention is paid to actual linguistic data and how appropriate they are for automatic parsing: linguistic data can be too complex for a given parser, morphological tags may not reflect well syntactic properties of words, a detailed, complex annotation scheme may be ill suited for automatic parsing.

In this paper, I present a study of this problem on the following case: automatic dependency parsing using the data of the Prague Dependency Treebank with two dependency parsers: MSTParser and MaltParser. I will show that by means of small, reversible simplifications of the text and of the annotation, a considerable improvement of parsing accuracy can be achieved.

In order to facilitate the task of language modeling performed by the parser, I reduce variability of lemmas and word forms in the text. I modify the system of morphological annotation to make it more suitable for parsing. Finally, the dependency annotation scheme is also partially modified. All such modifications are automatic and fully reversible: after the parsing is done, the original data and structures are automatically restored. With MaltParser, I achieve an 8.3% error rate reduction.

Keywords: dependency parsing; text simplification; syntax

1. Introduction

In dependency parsing, a slow progress in parsing reliability is achieved by new parsing algorithms and better feature settings. The actual linguistic data are much less analyzed. However, linguistic data and its annotation in dependency treebanks can be too complex for a given parser and better results can be achieved by simplifying data instead of developing new parsing algorithms.

In this paper, I present an experiment based on the data of Prague Dependency Treebank and two parsers, which has been extensively tested on these data: MSTParser and MaltParser. I use the settings of the parsers which achieved the best results. I automatically modify (simplify) training, test and new data on several levels: formal and lexical, morphological and syntactic. Each of these modifications slightly improves the accuracy of both parsers: MaltParser profits from these modifications more than MSTParser. At the end of the paper, I present some suggestions how to use this method for improving the parsing of other languages and treebanks.

2. Prague Dependency Treebank

The Prague Dependency Treebank (Hajič, 2006) is a corpus of Czech journalistic texts, manually annotated on several levels. It comprises 1.5 million tokens annotated on the level of surface syntax, the annotation being controlled by a special, very detailed and linguistically elaborated dependency scheme, based on the traditional Czech dependency syntax. In the PDT dependency syntactic structure of a given sentence, every token (including punctuation) depends either on another token, or on an external, artificial root of the dependency tree. Moreover, every token is assigned a surface syntactic function: basic (subject, object, attribute etc.), or auxiliary (auxiliary verb, preposition, coordinating conjunction etc.). Additionally, two kinds of non-dependency relations among elements of syntactic structure are distinguished, too: coordination and apposition.

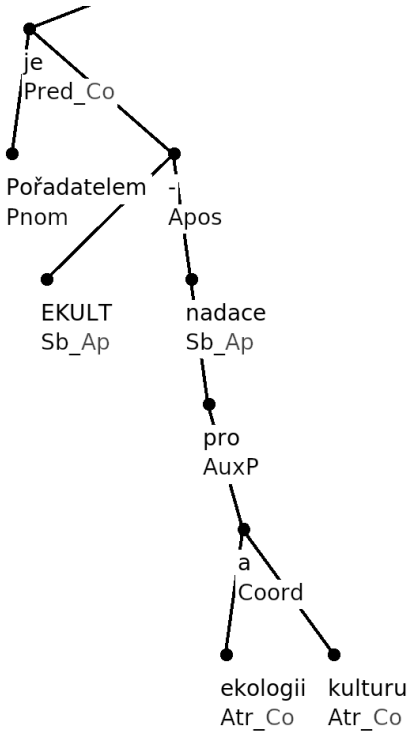

Figure 1: Example of a sentence structure in PDT.

Figure 1 shows a complex example of such a syntactic structure: *Pořadatelem je EKULT – nadace pro ekologii a kulturu* 'The organizer is EKULT – foundation for ecology and culture'. The syntactic functions *Pred* (predicate of the main clause), *Pnom* (nominal part of the verbonominal predicate), *Sb* (subject) and *Atr* (attribute) are basic functions; *AuxP* (preposition), *Apos* (apposition) and *Coord* (coordination) are auxiliary functions. The suffixes *_Co* and *_Ap* denote nodes in coordination or apposition relation.

2.1. Data cleanup

I used the version 2.5 of PDT (since then, PDT version 3.0 has been published), i.e. data which have been used for 10 years and thoroughly tested and revised. However, when I tested a rule-based module that focused on some specific linguistic phenomena, I noticed that some structures were annotated inconsistently, for example the word líto 'sorry' in structures as *je mi líto* 'I am sorry' was annotated in 6 occurrences as *Pnom*, in 3 as *Adv*. To correct these discrepancies, I devised a simple rule-based automatic tool to find probably incorrect structures and incorrect syntactic annotation. Automatically identified suspicious sentences were then manually revised by an independent linguist (not the author of this paper). Only about 0.25% of nodes were corrected. These corrections will be hopefully incorporated in a later version of PDT.

3. Dependency parsers

I experimented with two parsers yielding the best published results on PDT: MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2005). I used the settings which achieved the best accuracy on Czech (both unpublished). I could perhaps have obtained better results by tweaking these settings after every modification of data, but my aim was primarily to improve parsing results by modifying training and test data. More parsers will be tested in the near future.

Both parsers tested use automatically morphologically annotated text for training and for parsing.

3.1. MaltParser

The best results for MaltParser to my knowledge have been achieved by Daniel Zeman (86.1% UAS / 79.8% LAS) with LibSVMlearner and stacklazy algorithm. The parser uses all the values of the positional morphological tag (POS, number, gender, case etc.) as single features. Both forms and lemmas are available to the parser (which uses CoNLL format of data).

3.2. MSTParser

The best results for the MSTParser were obtained by Miroslav Spousta (85.9% UAS / 78.8% LAS). The parser uses only reduced morphological tags: POS and case or POS and subtype of POS (for POS without case) as they were proposed for Czech parsing by Collins et al. (1999). Wider tags (including, e.g., gender and number) paradoxically decrease the parser's accuracy. A simple MST format is used, where either forms, or lemmas can be made available to the parser (the original setting uses forms, I use lemmas in some of the experiments).

4. Data simplification

Syntactically annotated language data in a treebank are very complex, tens of thousands of different word forms and lemmas (in the case of PDT) combine with a detailed syntactic annotation, which uses tens of possible syntactic functions and a great variety of dependency relations (including non-projective dependency relations). Even for a human annotator, to annotate a new text according to a 300-page annotation manual is demanding. For stochastic parsers, such a task is even more challenging, as they do not 'understand' the text and syntactic and semantic relations between words. The parsers have to guess what information contained in the annotated text is important and reliable, and what information it can discard. Often, it guesses correctly, but the complexity of the data necessarily leads to errors.

The task of the parsers can be effectively made easier by simplifying language data, by training parsers to work with a simplified text and by parsing new, simplified text. After the parsing is done, all information discarded before parsing can be restored from a backup file. The modifications are performed on several levels: my software tool simplifies forms and lemmas, modifies morphological annotation, changes multiword expressions, and simplifies syntactic structure.

4.1. Reduction of forms and lemmas variability

To parse correctly, parsers cannot rely on morphology (POS, case, gender etc.) only, they need to use forms or lemmas of words. In the necessarily limited training data, however, it can only encounter a small fraction of lemmas of a given language, a large part of the words the parser encounters in a new text are 'out-of-vocabulary'. Moreover, even if a word does occur in the training data, most of them will occur only once or twice, which is not enough to base a reliable language model on. It is possible to partially remedy these shortcomings of natural language data by replacing words with identical syntactic properties by one representative. Consider following two sentences:

Pět lidí jelo s Klementem Gottwaldem za Stalinem do Moskvy.

'Five people went with Klement Gottwald to meet Stalin in Moscow.'

Sedm lidí jelo s Václavem Havlem za Walesou do Varšavy.

'Seven people went with Václav Havel to meet Walesa in Warsaw.'

Here, both sentences have the same syntactic structure and morphological properties, but half of the words differ in forms and lemmas. In my experiment, I define groups of words with identical syntactic properties and replace them by one representative for each group, for example

numbers as *five*, *six*, *seven*, *eight* ... are replaced by *five*; first names as *Klement*, *Václav*, *Vladimír*, *Jan* ... are replaced by *Jan*; names of cities like *Warsaw*, *Moscow*, *Prague*, *Paris* ... are replaced by *Prague* and so on. Both previously presented sentences will be transformed to the following sentence:

Pět lidí jelo s Janem Novákem za Novákem do Prahy.

'Five people went with Jan Novák to meet Novák in Prague.'

On the whole, members of some 80 groups of words are found and replaced, lowering the number of different forms in the training data by 19%. More details of experiments with forms and lemmas simplification can be found in (Jelínek, 2013).

4.1.1. Data sources for the simplification of forms and lemmas

Most of the lists of words with the same syntactic properties used in the data simplification method were taken from existing language databases. Many such lists are used in a project of rule-based morphological disambiguation (Petkevič, 2006). Some properties of verbs were derived from a valency dictionary: PDT-VALLEX (Urešová, 2009).

A few lists of words were created manually using a billion-word corpus of written Czech: corpus SYN (www.korpus.cz).

4.2. Adapting morphological annotation to parsing

The morphological annotation used in PDT (Hajič, 2004) is based on traditional Czech morphology, it was not designed for the purposes of parsing. It is adequate for some categories, but other categories are ill suited for parsing, they either contain too much information, or are confusing for the parser. Pronouns, for example, are divided into many semantic subclasses which do not reflect syntactic properties: the subcategory of indefinite pronouns contains both words as *někdo* 'someone' and *něco* 'something' which act as syntactic nouns in a sentence (they mostly have the function of subject or object), and words as *nějaký* 'some'*, jakýsi* 'a kind of' which act as syntactic adjectives (usually having the function of attribute). The morphological annotation can be automatically modified to remove irrelevant morphological information and add new distinctions important for parsing, as is the distinction between syntactic nouns and syntactic adjectives.

4.3. Simplification of syntactic annotation

The system of surface syntactic annotation in PDT is detailed and relatively complicated. In the treebank, 84 different syntactic functions are used (84 combinations of basic syntactic functions and the suffix for coordination / apposition, e.g. *Sb*, *Obj*, *Obj_Co*, *Atr_Ap*), 17 such combinations occur less than 10 times in all the data (the most frequent function, *Atr*, occurs 389 000 times).

For the parser, it is impossible to learn anything from the data about such rare cases. Therefore, it is possible to improve parsing results by lowering the number of syntactic functions in the training data: a part of these simplifications can be restored after parsing. In my experiment, for example, I completely removed combined syntactic functions (*AtrAdv*, see below) and I unified the suffixes for coordination and apposition.

4.3.1. Combined syntactic functions

In PDT, combined syntactic functions like *AtrAdv* or *ObjAtr* are used to express structural ambiguity. Such functions are not widely used in the data, therefore the parser is unable to learn to use them correctly: when parsing test data, the parser fails to identify these functions in 90% – 100% cases. Completely removing these functions from the data makes it impossible for the parser to use them (we lose these 0% – 10% correctly identified combined functions), but the parser cannot use these functions in the wrong context and it has fewer functions to learn.

4.3.2. Suffixes of syntactic functions

The use of suffixes *_Co* (coordination) and *_Ap* (apposition) is fully determined by a superordinate node (*Coord*, *Apos*), so it is possible to unify these suffixes for training and parsing and decrease in this way the number of functions (combinations) the parser has to work with. The suffixes can be restored using a rule-based module after the parsing is done. This simplification also slightly improves parsing results.

4.4. Simplification of multiword expressions

Another way of simplifying language data for the parser is the modification of multiword expressions and, more generally, of frequent expressions with a fixed syntactic structure. There is one condition though: only one node in this structure can be further determined by other nodes. Such a structure can be replaced by one token, a representative of the whole expression. In the post-processing phase, the original expression will be restored with its syntactic structure (which is known).

A typical example of such multiword expressions in PDT are (some) compound prepositions. The annotation manual lists some 70 compound prepositions, but only a few are systematically annotated as such in the data, for example *na základě* 'based on', *v rámci* 'within, in the frame of'. Morphologically, these prepositions are composed of a preposition and a noun, and parsers tend to treat them as other cases of prepositional phrases. These expressions are then replaced by one representative of the whole structure, e.g. *na_základě, v_rámci*, in both training and new data, and syntactic structure in the training data is modified appropriately. This way, the parser cannot make mistakes in such a structure. After the parsing is done, the structure is correctly restored.

I treat similarly some named entities such as *Frankturt nad Mohanem* 'Frankfurt am Main', *Ústí nad Labem* 'Ústí upon Elbe'. There are a few problematic co-occurrences of punctuation and words, which benefit from this approach, too. In Czech, the conjunction *-li* 'if' is attached

to the verb, as in *Je-li podnik v likvidaci, musí se dražit.* 'If the company is in liquidation, it must be auctioned.' Tokenization divides *Je-li* into three elements *Je* (verb 'to be'), - (hyphen) and *li* (conjunction). The presence of punctuation between the verb and the conjunction seems to confuse the parsers, because they parse sentences with this special conjunction much worse than others. If parsers work with - and *li* as one unit, they resolve these structures more easily.

4.5. Rule-based modifications of syntactic structures

For several syntactic phenomena, a more complex, rule-based modifications of syntactic annotation or structure were necessary. In Czech syntax, for example, no complement of the verb *být* 'to be' can have the syntactic function of object (Obj). Several special constructions in PDT are, however, annotated using the function *Obj* dependent on the verb *být*, for example *je mi/Obj líto* 'I am sorry' and *je mu/Obj 30 let* 'he is 30 years old'. These special cases are enumerated in the annotation manual and are easy to understand for a linguist. For stochastic parsers, it is confusing. When creating its language model, the parser learns that it is actually possible to assign the function *Obj* to a node dependent on the verb *být* and it then erroneously assigns this function on many other, improper occasions.

Since the occasions when *Obj* can be assigned in this context are well defined, we can completely eliminate all these cases in the training data, and thus we simplify language modelling for the parser and we can restore these functions by a rule-based module after the parsing is done.

5. Results

Many experiments with simplifications and modifications of data were tried. In two tables, I present a brief summary of the tests, performed with MaltParser and MSTParser on d-test data of PDT 2.5. I present both unlabeled attachment score (UAS) and labeled attachment score (LAS). The modifications of parsing and test data are incremental (modified data from a previous experiment are used in the subsequent one). In the first step, I present the results obtained with original data and parser settings (by D. Zeman / M. Spousta). In the following one, the same experiments are performed on the data with minor data corrections. The next step presents experiments with data with the simplification of word forms and lemmas. Morphological annotation better adapted to parsing is used on this data. The final step includes the modifications of syntactic annotation and of multiword expressions.

Sometimes, the increase in parser accuracy is below the statistical error, but all the modifications on every level have been tested several times with or without other changes, and they consistently improved parser's accuracy. Not all data modifications, however, resulted in accuracy increase. Some simplifications of syntactic structure, for example, even decreased parsers performance, therefore they were discarded from the simplification process.

5.1. MaltParser results

Table 1 presents the results (UAS / LAS) of MaltParser with various levels of data simplifications on d-test data of PDT 2.5.

MaltParser	*UAS / LAS*
original setting	86.11% / 79.80%
data clean-up	86.12% / 79.86%
simplification of forms and lemmas	86.44% / 80.58%
modification of morph. annotation	86.79% / 81.07%
modification of syntactic annotation	86.90% / 81.35%

Table 1: Results of the experiments with MaltParser

The largest contribution to accuracy increase in the experiment was brought by the simplification of word forms and lemmas (reduction of forms / lemmas variability by approx. 20%). All other modifications led to a much smaller increase.

5.2. MSTParser results

Table 2 presents the accuracy of MSTParser with various levels of data simplifications on d-test data of PDT 2.5.

MSTParser	*UAS / LAS*
original setting	85.93% / 78.80%
data clean-up	85.94% / 78.83%
simplification of forms and lemmas	86.12% / 79.00%
modification of morph. annotation	86.29% / 79.25%
modification of syntactic annotation	86.31% / 79.45%

Table 2: Results of the experiments with MSTParser

MSTParser benefits much less from the presented method than MaltParser. The increase in accuracy of MSTParser in most of the experiments is two (UAS) or four times (LAS) lower.

6. Portability to other languages

The procedure presented in the paper can be easily adapted to other languages and treebanks. The aim is to simplify language data as much as possible, without changing anything important for parsing. The best way is to use existing language resources, for example lists of words that are semantically similar and for which same syntactic properties can be expected.

6.1. Simplification of forms and lemmas

Most of these groups of words will be found among nouns. For example all female first nouns (*Mary*, *Susan*, *Helen*) can be found and replaced by one substitute (e.g.

Mary), and similarly for names of towns, states, rivers, months, days of the week and so on. Some groups of adjectives with the same syntactic properties can be probably found as well (e.g. *English*, *French*, *German*). A good target for the simplification of forms and lemmas are numerals: most of the subclasses of numerals (cardinals, ordinals, fractions...) can be simplified, replacing a potentially unlimited number of forms and lemmas by a few (representing each of the subclasses).

6.2. Phonetic variants

If your parser settings use word forms, you should also replace all the variants of words influenced by phonetic context by their basic form (e.g. *a* / *an*; in Czech, for example, two variants of the preposition *v* 'in' exist: *v* and *ve*, the variant *ve* being used when it is followed by a word starting with consonants *v* / *f* or a consonant cluster: *v Praze* / *ve Varšavě*: 'in Prague' / 'in Warsaw'; the form *ve* will be replaced by *v*).

6.3. Modification of morphological annotation

Moreover, it should be analysed, to what extent your system of morphological tagging meets the needs of syntactic parsing. As shown above in my brief analysis of the morphological tags in PDT, such systems can be either too detailed, or lack information about important syntactic properties. An appropriate modification could improve the results of parsing.

6.4. Multiword expressions

If multiword expressions with always the same syntactic structure can be found in your treebank (a structure, where only the head node can be determined by other nodes), such as compound prepositions or idioms with unusual word order, you can also replace these expressions by one proxy node, and restore the original structure after the parsing. In this way, the parser will not have to learn these particular structures.

6.5. Software tool

Technically, a simple software tool should be created to modify language data. It should work in two modes: one for the training data, in which it can also modify syntactic structures (simplify it, when needed), and the other one for new and test data, when it can work only with a morphologically annotated and lemmatized text, and it has to backup all text changes, to be able to restore the text after the parsing is done. Correct functioning of the software tool can be tested by simplifying (modifying) all training and testing data and then restoring them: the original text must be preserved.

7. Conclusion

In this paper, I presented a simple method for the improvement of dependency parsing, without having to change the settings of parsers. The method consists in careful automatic simplification of data at multiple levels. Discarded linguistic information is stored in a backup file and is restored after parsing is done. In my experiment, I reduce the variability of lemmas and forms in the text, I adapt morphological annotation for the needs of parsing, I simplify multiword expressions and in some cases I modify syntactic structure. The most efficient part of this method has proved to simplifying of forms and lemmas. The method described in the paper is language-dependent, but it can easily be applied to other languages, especially when lists of words with identical syntactic properties are available.

8. Acknowledgements

This research was supported by grant no. 13-27184S of the Grant Agency of the Czech Republic.

9. References

Collins, M., Hajič, J., Ramshaw, L., Tillmann, C. (1992): A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the ACL*. College Park, MD, USA, pp. 505--512.

Hajič, J. (2004): Disambiguation of Rich Inflection, Computational Morphology of Czech. Karolinum, Charles University Press, Prague, Czech Republic.

Hajič, J. (2006): Complex Corpus Annotation: The Prague Dependency Treebank. In Šimková M. (Ed.), *Insight into the Slovak and Czech Corpus Linguistics*. Veda, Bratislava, Slovakia, pp. 54--73.

Jelínek, T. (2013): Improving Dependency Parsing by Filtering Linguistic Noise. In *Proceedings of the 16th International Conference Text, Speech, and Dialogue, TSD 2013*. LNCS, Springer Verlag, Heidelberg, Germany, pp. 288--294.

McDonald, R., Pereira, F., Ribarov, K., Hajic, J. (2005): Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT/EMNLP*, Vancouver, Canada, pp. 523--530.

Nivre, J., Hall, J., Nilsson, J. (2006): MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*. Genoa, Italy, pp. 2216--2219.

Petkevič, V. (2006): Reliable Morphological Disambiguation of Czech: Rule-Based Approach is Necessary. In: Šimková M. (Ed.), *Insight into the Slovak and Czech Corpus Linguistics*. Veda, Bratislava, Slovakia, pp. 26--44.

Urešová, Z. (2009): Building the PDT-VALLEX valency lexicon. In *On-line Proceedings of the fifth Corpus Linguistics Conference*, University of Liverpool, UK.