PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 7–14

Dependency Parsing as a Sequence Labeling Task

Drahomíra "johanka" Spoustová, Miroslav Spousta

Institute of Formal and Applied Linguistics, Charles University in Prague

Abstract

The aim of this paper is to explore the feasibility of solving the dependency parsing problem using sequence labeling tools. We introduce an algorithm to transform a dependency tree into a tag sequence suitable for a sequence labeling algorithm and evaluate several parameter settings on the standard treebank data. We focus mainly on Czech, as a high-inflective free-word-order language, which is not so easy to parse using traditional techniques, but we also test our approach on English for comparison.

1. Introduction

Dependency parsing became very popular in the recent years and many different algorithms were suggested and evaluated for this task.

Dependency structure is a rooted tree where each node (except the root) corresponds to one word of the underlying sentence. The dependency relation between two nodes (parent and child) is captured by an edge between these two nodes. The actual type of the relation is given as a function label of the edge.

The aim of a dependency parser is to assign correct parent node index to each child node, optionally also with the dependency relationship label. The standard method of evaluating dependency parser accuracy is computing the percentage of children that got the correct parent index (and optionally the dependency relationship label), among all words in a test data set.

Sequence labeling is a process where each token in a sentence is assigned a label from a fixed set. Common examples include Part-of-Speech (POS) tagging, Named

Corresponding author: johanka@ucw.cz
Cite as: Drahomíra "johanka" Spoustová, Miroslav Spousta. Dependency Parsing as a Sequence Labeling Task. The Prague Bulletin of Mathematical Linguistics No. 94, 2010, pp. 7–14.
doi: 10.2478/v10108-010-0017-3.

entity recognition and Semantic Role Labeling tasks, where we label every word with a tag. There are many well-studied algorithms that proved to be successful, many of them based on the Hidden Markov models and the Viterbi algorithm, such as Conditional Random Fields (Lafferty et al., 2001) or Averaged Perceptron (Collins, 2002).

In the area of parsing, the sequence labeling techniques were applied mainly to the NP-chunking (shallow parsing) task.

In the following sections, we would like to explore the possibility to turn the dependency parsing into a sequence labeling task.

Our approach is somewhat similar to LTAG supertagging introduced by (Bangalore and Joshi, 1999), but their approach mas been deeply explored mainly for English and cannot be directly applied to free-word-order non-projective languages with rich inflection, like Czech.

The paper is organized as follows: Section 2 introduces the data used for our experiments. In section 3, we propose a tagset and conversion algorithm for the sequence labeling task and evaluate the performance of selected settings of the proposed algorithm on the English and Czech data sets. Section 4 presents current results and concludes.

2. The data

In order to demonstrate that our approach is not restricted to a specific language, we performed our data conversion experiments on two languages with completely different morphological characteristics and tagsets, English and Czech.

For English, we used CoNLL Shared Task 2009 data, which is a dependency representation of (a part of) the Penn Treebank (Marcus et al., 1994). We used the columns PLEMMA, PPOS (automatically assigned part-of-speech tag and morphological lemma), HEAD and DEPREL (manually annotated labeled dependency relationship). For Czech, we selected the corresponding data (i.e. manual annotation of the dependency relationships and automatic POS tagging) from the Prague Dependency Treebank 2.0 (Hajič et al., 2006), analytical layer. The Czech morphological tagset is described in (Hajič, 2004).

Main characteristics of the data[1] can be found in Table 1.

3. The data conversion algorithm

Standard dependency tree representation assigns a parent node index to every word in a sentence. The process of turning dependency parsing into a sequence la-

[1] All experiments were performed using only the Train and the Development data sets. We save the Evaluation data set for the final evaluation of the (hypothetical) parser.

Table 1. The data characteristics

data set	tokens	sentences	tagset size
English train	958,167	39,278	46
English dev	33,368	1,335	45
Czech train	1,172,299	68,562	1330
Czech dev	158,962	9,270	1041

Table 2. The example sentence

	FORM	PLEMMA	PPOS	HEAD	DEPREL	DEPTAG
1	The	the	DT	4	NMOD	NN_R3 NMOD
2	luxury	luxury	NN	3	NMOD	NN_R1 NMOD
3	auto	auto	NN	4	NMOD	NN_R1 NMOD
4	maker	maker	NN	7	SBJ	VBD_R1 SBJ
5	last	last	JJ	6	NMOD	NN_R1 NMOD
6	year	year	NN	7	TMP	VBD_R1 TMP
7	sold	sell	VBD	0	ROOT	00_00 ROOT
8	1,214	1,214	CD	9	NMOD	NNS_R1 NMOD
9	cars	car	NNS	7	OBJ	VBD_L1 OBJ
10	in	in	IN	7	LOC	VBD_L1 LOC
11	the	the	DT	12	NMOD	NNP_R1 NMOD
12	U.S.	u.s.	NNP	10	PMOD	IN_L1 PMOD

beling task must turn this "relative" information into an absolute label, which is to be assigned to the words. Let us call this label *deptag*[2].

We designed the *deptag* to contain the following parts:

1. parent node's POS tag
2. additional information needed to decide between more possible parent-candidates with the same POS tag: direction and distance[3].
3. The DEPREL label is added to the *deptag* (no transformation is needed here) to be also predictable by the sequence labeling algorithm.

An example sentence transformation can be found in Table 2.

[2]Unfortunately, the word *supertag* was already taken in (Bangalore and Joshi, 1999).

[3]We use L (R) for left (right) direction, respectively, and a positive number as a distance marker. E.g. VB_R2 means "second VB token on the right" (i. e. the first VB candidate on the right is omitted)

3.1. Tagset reduction

Naturally, a too large *deptagset* (tagset of the deptags) leads to the data sparseness problem. It turns out that the size of the tagset may be reduced by adjusting two parameters of the algorithm:

1. Considering only a fragment of the full POS tag during the *deptag* construction. This is mainly useful for a rich POS-tag set language (such as Czech with more than 4000 possible POS-tags and only 1330 of them appearing in the training data) and is much less important for English (with only 46 tags). We tried to reduce the 15-positions tagset to 5 or 2 most important positions for Czech[4].
   At this point, the assigned *deptag* corresponds to exactly one parent node. In other words, the transformation from the tree into a *deptag* sequence remains lossless.
2. Constraining maximum *distance* to reduce the *deptagset*. Here the *distance* does not mean the absolute number of tokens between parent and its child, but the maximum number of possible candidates with the same POS tag to be distinguished. For example, if the maximum is set to 3, then all third, fourth and further candidates will be labeled with 3 and translated back to the third candidate.
   This constraint, however, makes the transformations lossy.

Let us define the *transformation error rate* of the selected algorithm configuration as following: For each node n let p^1_n be the parent node index assigned to the node n in the source dependency tree, D_n a deptag, to which p^1_n is translated using the selected algorithm configuration, and p^2_n a parent node index, to which the the deptag D_n is translated back (the process of translating back is unambiguous). *Transformation error rate* is the percentage of nodes, where $p^2_n \neq p^1_n$.

Table 3 shows the *deptagset* size (labeled, unlabeled) and transformation error rates for a few selected configurations. Considering usual performance of dependency parsers lying between 80 % and 90 %, setting the *maximum distance* to 3 introduces only a small decrease of performance (lower than 1% transformation error rate).

3.2. The list of possibilities

Usually, the sequence labeling algorithm chooses one label from a list of plausible labels for every token. There are several methods how to generate such a list. We can compare the methods for generating the list of possible deptags using a standard measure: precision and recall.

[4] 5 — Part of speech, Detailed part of speech, Gender, Number, Case; 2 (originating from (Collins et al., 1999)) — first letter is the main POS, second letter is the Case field if the main POS is the one that displays case, while otherwise the second letter is the detailed POS.

Table 3. deptagset *size (labeled, unlabeled) and transformation error rate. (POS - Part-of-speech tag size, max - maximum distance.)*

language	POS	max. distance	tagset size		error
			labeled	unlabeled	
en	full	-	1523	244	0.00
en	full	3	1305	159	0.52
en	full	1	809	78	7.56
cz	full	-	10449	1979	0.00
cz	full	3	9949	1882	0.30
cz	full	1	8679	1601	2.29
cz	5	-	7820	1319	0.00
cz	5	3	7299	1214	0.30
cz	5	1	6016	958	2.45
cz	2	-	3760	389	0.00
cz	2	3	3127	244	0.36
cz	2	1	2054	122	3.32

In order to enable sequence labeling algorithm to work well on our transformed data, we aim to keep recall as high as possible, while introducing a reasonable precision.

- To achieve 100% recall, we need to consider every token in the sentence to be possible parent of all other tokens. I.e. in a n-token (incl. root) sentence, every token has $n-1$ possible parents, thus translated into $n-1$ unlabeled *deptags*. Every *deptag* can (theoretically) be combined with every deprel label. Let dep be size of the set of the deprel labels. So the final list of possibilities contains $(n-1) * dep$ labeled deptags for every token.
  This approach achieves 100% recall, but very poor precision.
- The list of possible combinations *deptag* + dependency relationship label (*labeled deptags*) can be simply derived from the training data. To avoid the data sparseness problem, we choose to discriminate the tokens only through their morphological tags, i.e. to ignore their form and lemma. For example, if a token has the *NNP* POS tag, we add to its list of possible *labeled deptags* manually annotated parents (translated into the *labeled deptags*) of all *NNP* tokens found in the training data.
  This approach achieves 100% recall only for the training data. Its precision is much better than the precision of previous approach, but still remains low.
- Precision of the previous approach can be increased by restricting the maximum "length of relationship" (in absolute number of tokens) between parent and child. We omit every possible relationship which is "longer" (in terms of

Table 4. Generating a labeled list of possibilities; recall2 = recall including conversion back to the tree representation

basic options	list options	precision	recall	recall2
en full -	full	0.06	100	100
en full 3	full	0.06	100	99.48
en full 3	train	1.42	99.77	99.26
en full 3	train+length	2.45	99.43	98.93
cz 5 3	full	0.05	100	99.70
cz 5 3	train	1.28	98.21	97.93
cz 5 3	train+length	2.28	96.86	96.58
cz 2 3	full	0.05	100	99.64
cz 2 3	train	0.82	99.53	99.18
cz 2 3	train+length	1.46	99.01	98.67

absolute value of the subtraction between the nodes indexes) than the longest relationship seen in the training data between nodes with the same POS tags as the ones of the considered tokens.

Precision and recall of selected variants of the approaches described above can be found in Tables 4 (labeled) and 5 (unlabeled), as measured on the development data set.

4. Results and Conclusion

We have shown that our data conversion algorithm can fully represent a labeled dependency tree, both for a rich morphology language with large tagset and for a language with very small tagset.

The parameters of the conversion can be theoretically set in the manner that keeps the conversion absolutely lossless. We have proposed various kinds of (slightly lossy) reductions of the solution space.

As we have proposed in the introduction section, this article focuses on the data conversion algorithm, i.e. data preparation for the sequence labeling algorithm.

As a proof-of-concept and possible baseline, we have processed the converted data with the averaged perceptron algorithm (Collins, 2002) with trivial trigram feature set, the results (accuracy of the final labeled tree) varied (depending on configuration) between 5-10 % below state-of-the-art (according to CoNLL Shared Task 2009 results and the `http://ufal.mff.cuni.cz/czech-parsing` page). Additional up to about 1 % will be loosed if we use as a postprocessing some sort of greedy algorithm which will fix the cycles to ensure the result is tree.

Our approach can thus be used as a simple and fast way to build an "approximative" parser in case there is no better solution available (e.g. due to license restrictions).

Table 5. Generating an unlabeled list of possibilities; recall2 = recall including conversion back to the tree representation

basic options	list options	precision	recall	recall2
en full -	full	3.24	100	100
en full 3	full	3.52	100	99.48
en full 3	train	4.80	99.92	99.39
en full 3	train+length	6.68	99.75	99.24
cz 5 3	full	4.17	100	99.70
cz 5 3	train	6.52	99.08	98.79
cz 5 3	train+length	9.76	98.20	97.91
cz 2 3	full	4.27	100	99.64
cz 2 3	train	5.18	99.91	99.55
cz 2 3	train+length	7.26	99.67	99.32

The performance can be further improved by selection of the sequence labeling algorithm and its configuration (such as feature set selection).

Acknowledgments

The research described here was supported by the project *GA405/09/0278* of the Grant Agency of the Czech Republic.

Bibliography

Bangalore, Srinivas and Aravind K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265, 1999.

Collins, Michael. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8, Philadelphia, PA, 2002.

Collins, Michael, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, College Park, Maryland, USA, June 1999. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P99-1065`.

Hajič, Jan. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague, 2004. ISBN 80-246-0282-2.

Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. Prague Dependency Treebank v2.0, CDROM, LDC Cat. No. LDC2006T01. Linguistic Data Consortium, Philadelphia, PA, 2006.

Lafferty, John, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 2001.

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994. ISSN 0891-2017.

Address for correspondence:
Drahomíra "johanka" Spoustová
johanka@ucw.cz
Institute of Formal and Applied Linguistics
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic