Parsing Aided by Intra-Clausal Coordination Detection

Domen Marinčič, Matjaž Gams, Tomaž Šef
Jozef Stefan Institute, Ljubljana
Department of Intelligent Systems

Zdeněk Žabokrtský
Charles University in Prague
Institute of Formal and Applied Linguistics

Abstract

We present an algorithm for parsing with detection of intra-clausal coordinations. The algorithm is based on machine learning techniques and helps to decompose a large parsing problem into several smaller ones. Its performance was tested on Slovene Dependency Treebank. Used together with the maximum spanning tree parsing algorithm it improved parsing accuracy.

1 Introduction

One of the problems syntactic parsers are facing is a successful processing of coordinations. A promising approach to deal with coordinations is described in [5]. They use tree transformations and report the performance improvement of MaltParser, while their method did not prove successful using the maximum spanning tree (MST) parser [4]. Another approach, described in [2] is limited to nominal coordinations since it is based on the semantic similarity of conjoined nouns.

We propose an *a*lgorithm for *p*arsing with intr*a*-clausal *c*oordination *d*etection (APACD) that uses machine learning (ML) techniques. It is based on the presumption that intra-clausal coordinations, represented as subtrees in the sentence tree, can first be reduced to meta nodes. Coordinations and sentences with meta nodes can then be parsed separately. Finally, the complete parse can be obtained by merging the resulting trees. This way a large parsing problem can be decomposed into several smaller ones. APACD was tested on Slovene Dependency Treebank, (SDT, ≈ 38.000 tokens) [1]. Gold POS-tags from the treebank were used in the training and the testing phase.

De Smedt, K., Hajič, J. and Kübler, S. (Eds.)
Proceedings of the Sixth International Workshop
on Treebanks and Linguistic Theories (2007)

2 Description of APACD

APACD works on prepositional, nominal and adjectival coordinations. It can only handle non-embedded coordinations. The algorithm consists of three steps:

1. Groups of head words. First, the groups of the head words of conjunct phrases, ordered by their appearance in the sentence, are formed (see Fig. 1a). The group has to comply with the following rules:

- All head words must have the same POS and case.[1]
- Between each pair of the head words there has to be a valid delimiter (a comma or a coordinating conjunction). The sentence is first split to the segments between the delimiters as proposed by [3]. A delimiter is valid if at least one of the surrounding segments contains no finite verbs.
- Tokens, not allowed between the head words: colons, semicolons, dashes, brackets, finite verbs, relative pronouns and subordinating conjunctions.

After applying these rules, we get a set of groups of head words which is further filtered by ML classifiers. For each group, the sequence of tokens (w_1,..., w_n) representing the group is converted to pairs of neighboring head words (w_i, w_{i+1}), $0 < i < n$. Each pair is then classified by a ML classifier. If at least one pair is classified negatively, the whole group is discarded.

Three separate classifiers are used, one for each POS of the head words APACD can handle. The adaboost algorithm with the J48 decision tree as the core classifier from WEKA [6] is used. The examples for training the classifiers were extracted from SDT. To describe the examples with the attributes, two sections of the tokens between the head words are formed. The section A consists of the tokens between the first head word and the delimiter (empty in Fig. 1). The section B consists of the tokens between the delimiter and the second head word (underlined in Fig. 1). The attributes are the following:

- the presence of a preposition/an adverb in the section (4 attributes, binary values),
- the presence of a noun/an adjective matching/non-matching with the head word in case, number and gender in the section (8 attributes, binary values),
- the number of words in the section (2 attributes, values: 0, 1, 2, >2),

[1] APACD does not handle the coordinations where the heads of conjuncts do not match this condition.

- class (1 attribute, binary values).

2. Meta nodes. For each group of the head words the previous step yields a sequence of tokens spanning from the leftmost to the rightmost head word (i.e., all the intermediate tokens from the original sentence are included). All the sequences are reduced to meta nodes being assigned the same POS and case as the corresponding head words (Fig. 1b).

3. Merging dependency trees. To get the main tree (Fig. 1c) the sentence with the meta nodes is parsed by the MST parser version 0.2 [4][2], currently the best for Slovene. From each subtree having a meta node as its root a new sequence of tokens is created (Fig. 1d). The sequence contains all the tokens reduced to the meta node plus the descendants of the meta node in the main tree. All the sequences are also parsed by the maximum spanning tree parser (Fig. 1e). To obtain the complete parse, the resulting trees representing the coordinations are inserted into the main tree substituting the meta node subtrees (Fig. 1f).

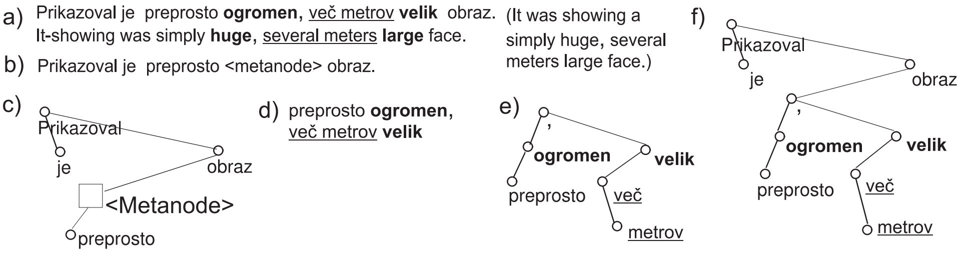
a) Prikazoval je preprosto ogromen, več metrov velik obraz.
It-showing was simply huge, several meters large face.
(It was showing a simply huge, several meters large face.)
b) Prikazoval je preprosto <metanode> obraz.
c)
Prikazoval
je
obraz
<Metanode>
preprosto
d) preprosto ogromen, več metrov velik
e)
,
ogromen
velik
preprosto
več
metrov
f)
Prikazoval
je
obraz
,
ogromen
velik
preprosto
več
metrov

Figure 1: An example showing how APACD processes a sentence with one group of two head words (boldface).

3 Evaluation of intra-clausal coordination retrieval

In our first experiment we estimated recall and precision of intra-clausal coordination retrieval. Following the precise definition adopted for this experiment, the intra-clausal coordination in SDT is the sequence of tokens of a subtree that matches these conditions:

- The grammatical function of the root of the subtree is 'Coord'.

- The root has at least two children (not counting the punctuation tokens and conjunctions) and none of them is a finite verb.

[2] The parser uses the maximum spanning tree algorithm over a weighted sentence graph to obtain the parse of a sentence.

In total, SDT contains 1456 intra-clausal coordination constructions, with the following distribution reflecting the conjunct head POS: (i) prepositions 6%, (ii) nouns 42%, (iii) adjectives 31%, (iv) other POS or mixed POS in one coordination 21%. The evaluation of APACD on the first three types, averaged from 10-fold cross validation on the SDT data (10 iterations with 90% of data for classifiers' training and 10% for testing in each iteration) is presented in Table 1. The fourth type is not considered in the measurements.

Coordination type	Prepositional	Nominal	Adjectival	All
Recall	60%	72%	81%	79%
Precision	69%	69%	95%	78%

Table 1: Recall and precision of intra-clausal coordination retrieval

Coordination detection can help the parsing process if the positive influence of constraining the parsing process is larger than the impact of causing additional errors by declaring wrong sequences of tokens as coordinations. Therefore, high precision is preferred, while high recall might not be crucial, since missing out some valid coordinations does not introduce new errors. At the first glance, the overall precision might seem quite low. However, the cases where only parts of coordinations were detected – which were treated as errors in our measurements – might still guide the parsing process towards better accuracy.

As expected, the best results were achieved on the least complex adjectival coordinations. Among false positives at nominal coordination retrieval, 32% were actually appositions. These cases should rather not be viewed as errors but as a positive contribution: appositions are represented as subtrees as well and the same mechanism can be applied to them as to intra-clausal coordinations. However, the problem of distinguishing nominal coordinations and appositions remains out of the scope of this paper.

The results for prepositional coordinations were the worst due to their high complexity compared to the other types of intra-clausal coordinations. The fourth type, containing the coordinations not covered by APACD, gives us some more room for improving the algorithm.

4 Evaluation of dependency parsing

In the final experiment we evaluated the complete APACD as described in Section 2 on the task of dependency parsing. 10-fold cross-validation on the data from SDT was used. Two distinct parsing models were trained for coordinations and main trees. Unlabeled attachment score (UAS) was measured. Another, plain MST parser without coordination detection achieved an unlabeled attachment score of 79.88%. This result served as the baseline.

Then, two series of tests were performed with APACD. The results are shown in Table 4. In the first series (2$^{\text{nd}}$ row), intra-clausal coordination retrieval was done only by applying the three rules, described in the first step of APACD in Section 2. No additional filtering with the ML classifiers was performed. In the second series the ML classifiers were included in the retrieval process (3$^{\text{rd}}$ row). Separate tests were run using various maximum allowed numbers of tokens in the sections A and B (1$^{\text{st}}$ row). Statistical significance of the results was estimated by the resampled t-test proposed by [6]. The results marked by * are better than the baseline at the 95% confidence level. The best result was achieved with the ML classifiers, limiting the maximum size of the sections A and B to 5 tokens, which is by 0.62 percentage points better that the baseline result.

Max. number of tokens allowed in sections A, B	4	5	6	7
UAS[%] without ML classifiers	80.29	80.43	80.43	80.39
UAS[%] with ML classifiers	80.41	*80.50	*80.48	*80.45

Table 2: Results of the experiment

Another test was performed to determine the upper bound of accuracy improvement. Instead of retrieving intra-clausal coordinations as described in the first step of APACD, SDT was used as an oracle, providing 100% accurate information about intra-clausal coordinations. The modified version of the algorithm achieved an unlabelled attachment score of 81.37%. This result can be regarded as the theoretical limit for the accuracy of APACD, if the intra-clausal coordination retrieval step were done perfectly.

5 Conclusion and future work

Our experiments have shown that decomposing large parsing problems to smaller ones is beneficial in terms of improving overall parsing accuracy. Considering the statistically significant improvement of 0.62 percentage points one should keep in mind that APACD focuses on a single syntactic phenomenon – intra-clausal coordinations, which only appear in 30% of all sentences of SDT. Since the time complexity of the intra-clausal coordination retrieval step is $O(n)$, n being the number of tokens in the sentence, additional time consumption is acceptable. Furthermore, with the experiments we have shown how to make use of the additional information provided by the richly inflected languages for improving parsing results.

In general, the reduction mechanism enforces projectivity. However, this is not an issue for APACD: no non-projective edges go out from correctly detected intra-clausal coordinations since they are represented by closed subtrees.

There are further ways how to improve APACD. One of the current problems is the rigid treatment of coordinations. APACD either declares a sequence of tokens a coordination or not. It would probably be better to raise the weights of the appropriate edges in the sentence graph and let the MST algorithm find the best solution. Another improvement would be to allow for searching embedded intra-clausal coordinations. Further, the context left of the leftmost head word and right of the rightmost head word of the coordination could be included into the attribute model of the ML classifiers. A possible direction of future research is to use the information provided by intra-clausal coordination detection for clause splitting: the commas and the conjunctions inside coordinations are *not* the candidates for clause borders.

Acknowledgments

This work was supported by the Slovenian Ministry of Higher Education, Science and Technology, and by the project MSM 0021620838 of the Ministry of Education of the Czech Republic.

References

[1] S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtský, A. Žele (2006). Towards a Slovene Dependency Treebank. *5th International Conference on Language Resources and Evaluation*, Genova, Italy.

[2] D. Hogan (2007). Empirical Measurements of Lexical Similarity in Noun Phrase Conjuncts. *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

[3] V. Kuboň, M. Lopatková, M. Plátek, P. Pognan (2006). Segmentation of Complex Sentences. *9th International Conference on text, speech and dialogue*, Brno, Czech Republic.

[4] R. McDonald, F. Pereira, K. Ribarov, J. Hajič (2005). Non-projective Dependency Parsing Using Spanning Tree Algorithms. *Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver, Canada.

[5] J. Nilsson, J. Nivre, J. Hall (2007). Generalizing Tree Transformations for Inductive Dependency Parsing. *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

[6] I. H. Witten, E. Frank (2005). Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition, Morgan Kaufmann, San Francisco.