Parsing Aided by Intra-Clausal Coordination Detection

Domen Marinčič, Matjaž Gams, Tomaž Šef, Jozef Stefan Institute, Ljubljana, Slovenia
\{domen.marincic, matjaz.gams, tomaz.sef\}@ijs.si

Zdeněk Žabokrtský
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
zabokrtsky@ufal.mff.cuni.cz

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 successful processing of coordinations. A promising approaches 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 and is based on the semantic similarity of conjoined nouns.

We propose an algorithm for parsing with intra-clausal coordination detection (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, \( \approx 38,000 \) tokens) [1]. Gold POS-tags from the treebank were used in the training and the testing phase.
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 conditions:

   - 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.
   - The members of the group \((w_1, \ldots, w_n)\) are split to the pairs of the neighboring head words \((w_i, w_{i+1}), 0 < i < n\). Each pair of the head words has to be classified positively by a ML classifier. The adaboost algorithm with the J48 decision tree as the core classifier from WEKA [6] is used. Three separate classifiers are used, one for each POS of the head words APACD can handle. 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:

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

¹APACD does not handle coordinations where the heads of conjuncts do not match this condition.
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. All such 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 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 such 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).

![Figure 1: An example sentence with one group of two head words (boldface) as processed by APACD.](image)

3 Evaluation of intra-clausal coordination retrieval

In our first experiment we estimated recall and accuracy of intra-clausal coordination retrieval, using the first step of APACD only. Following the precise definition adopted for this experiment, the intra-clausal coordination in SDT is a 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 none of them is a finite verb.

The experiment was performed using 10-fold cross-validation. In each iteration, 90% of SDT was used for training the ML classifiers, while the

---

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

³In total, SDT contains 1456 intra-clausal coordinations.
remaining 10% were searched for intra-clausal coordinations. The results of the measurements over the whole set of intra-clausal coordinations are presented in the sixth column in Table 1.

For a detailed analysis, intra-clausal coordinations were further split into four subsets: coordinations of (i) prepositional phrases, (ii) nominal phrases, (iii) adjectival phrases and (iv) other types of coordinations. The measurement of accuracy does not apply to the fourth subset, because APACD can only detect coordinations from the first three subsets. The results are presented in the columns two to five in Table 1.

<table>
<thead>
<tr>
<th>Coord. type (subset)</th>
<th>Prepositional</th>
<th>Nominal</th>
<th>Adjectival</th>
<th>Others</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of subset</td>
<td>6%</td>
<td>42%</td>
<td>31%</td>
<td>21%</td>
<td>100%</td>
</tr>
<tr>
<td>Recall</td>
<td>60%</td>
<td>72%</td>
<td>81%</td>
<td>0%</td>
<td>79%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>69%</td>
<td>69%</td>
<td>95%</td>
<td>-</td>
<td>78%</td>
</tr>
</tbody>
</table>

Table 1: Recall and accuracy 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 accuracy 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 accuracy might seem quite low. However, 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.

Among the measurements on the subsets, as expected, the best results were achieved on the least complex adjectival coordinations. When retrieving nominal coordinations, 32% of false positives 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 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 subset, containing 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 2. In the first series, intra-clausal coordination retrieval was done without the ML classifiers (2\textsuperscript{nd} row), while in the second series the ML classifiers were included in the retrieval process (3\textsuperscript{rd} row). We also experimented with the different maximum allowed number of tokens in the sections A and B (1\textsuperscript{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 than the baseline result.

<table>
<thead>
<tr>
<th>Max. number of tokens allowed in sections A, B</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAS[%] without ML classifiers</td>
<td>80.29</td>
<td>80.43</td>
<td>80.43</td>
<td>80.39</td>
</tr>
<tr>
<td>UAS[%] with ML classifiers</td>
<td>80.41 *</td>
<td>80.50 *</td>
<td>80.48 *</td>
<td>80.45 *</td>
</tr>
</tbody>
</table>

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, such as Slavic and Finno-Ugric 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 by definition closed sub-
trees.

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
detecting intra-clausal coordinations for clause splitting: the commas and
the conjunctions inside the 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

(2006). Towards a Slovene Dependency Treebank. 5th International
Conference on Language Resources and Evaluation, Genova, Italy.

Phrase Conjuncts. 45th Annual Meeting of the Association for Compu-
tational Linguistics, Prague, Czech Republic.

of Complex Sentences. 9th International Conference on text, speech and
dialogue, Brno, Czech Republic.

Dependency Parsing Using Spanning Tree Algorithms. Joint Confer-
ence on Human Language Technology and Empirical Methods in Natu-
ral Language Processing, Vancouver, Canada.

for Inductive Dependency Parsing. 45th Annual Meeting of the Associa-
tion for Computational Linguistics, Prague, Czech Republic.