IJCLA VOL. 5, NO. 1, JAN-JUN 2014, PP. 117–128
RECEIVED 07/01/14 ACCEPTED 06/02/14 FINAL 17/06/14

Automatic Recognition of Clauses

OLDŘICH KRŮZA AND VLADISLAV KUBOŇ

Charles University in Prague, Czech Republic

ABSTRACT

This paper describes an attempt to solve the problem of recognizing clauses and their mutual relationship in complex Czech sentences on the basis of limited information, namely the information obtained by morphological analysis only. The method described in this paper may be used in the future for splitting the parsing process into two phases, namely (1) Recognizing clauses and their mutual relationships; and (2) Parsing the individual clauses. This approach should be able to improve the result of parsing long complex sentences.

1 INTRODUCTION

Despite the progress achieved in recent years in the field of natural language parsing it still makes sense to seek alternative approaches to the problem. Parsing is a procedure *performed by a human or a computer* whose input are words of a sentence, typically with morphological annotation, and whose output is a syntactical structure on these words, in our case represented by a dependency tree. Clearly, a sentence consists of words, just like a human body consists of cells. Nobody sane would describe a human body as a mere cluster of cells though. A body apparently consists of organs, and those consist of cells. Even if a sentence can hardly have 10^{14} words[1], we believe the relation between words, clauses and a sentence is similar to that of cells, organs and a body.

Clauses are in many aspects autonomous and many grammatical relationships are either limited within a clause, or are among clauses as atomic units. For example, Petkevič [1] states that

[1] This is roughly how many cells a human body has.

- word order,
- valency and valency scopes, adjunct scopes, scopes of modifications (local, temporal, concessive, etc.),
- agreement and its scope,
- analytical predicate and its scope,
- constituent coordination and its scope,
- internal coreference/anaphor

are all intra-clausal relationships.

In our approach we suggest to step aside from the traditional path of performing full-fledged parsing immediatelly after morphological tagging of individual word forms and to attempt to detect the organs of complex sentences, individual clauses, first, and to determine their mutual relationships prior to the actual parsing phase. The next phse could then, subsequently, concentrate on parsing individual clauses one by one. Not only does this more refined sequence of steps make a lot more linguistic sense, but it also decreases the number of words a parser has to consider at a time and thus simplifies the task. We hope that this approach could also speed up the whole procedure and raise the overall efficiency.

2 Building a Model for Clause Recognition

Previous work on clause detection has focused on English and was mostly limited to finding clause boundaries, not their dependency relations like we do. One occasion of focus to clause identification was the CoNLL-2001 shared task [2, 3]. Meanwhile, Prague Depencency Treebank in its version 2.5 contains data that mark up clauses based on the research on sentence *segments* [4]. The clauses have been annotated by an automatic procedure that uses the surface syntax layer to identify the clause that each token belongs to [5]. This approach differs from the one presented in this paper in the type of information used. It exploits the information from already analyzed sentences and their surface syntactic trees and thus it cannot be used for splitting the parsing process into more phases. On the other hand, our approach aims at using the morphological information only for automatic clause identification.

The task of automated clause recognition is not trivial from the programming point of view. We'd like to use a statistical approach using machine learning. However, the task is not easily mapped to standard regression or classification that machine learning has standard procedures for. The output of clause recognition should include:

1. the set of clauses,
2. the dependency, and coordination relations among clauses,
3. distribution of words to individual

clauses.

2.1 *Incremental Approach*

Our first attempt at solving the task was an incremental solution, where sub-tasks would be defined, each solvable as a classification or regression task. The final result would then be obtained by chaining the subtasks.

The first step was the identification of clause borders. We trained a simple discriminative model based on features that drew on previous research of sentence segments described in [6]. Segments happen to be mostly bordered by conjunctions or punctuation in Czech and we expolited this. We have used the following features for our clause boundary model:

- word's lemma,
- whether the right neighbor is a *separator*,
- whether the left neighbor is a separator,
- whether the left neighbor is a conjunction.

A *separator* is a punctuation mark or one of the coordinate conjuntions that do not enforce a comma (*a*, *nebo*, *ani*, *i*).

The set of lemmata was limited to a small set of those that represent conjunctions, punctuation, and single-letter words.

These features were chosen so that they use all the morphological information that the surface-syntax-tree-to-clauses converter described in Krůza, Kuboň 2009 uses. This converter made it possible to use the data from the analytical layer of the Prague Dependency Treebank as training data.

The predicted feature was whether a word is at the boundary of a clause. Specifically, whether it is the starting word of a clause, whether it is the final word of a clause, whether this word is just before the start of a contained sub-clause and whether this word is just after the end of a contained sub-clause.

Despite its simplicity, this model achieved about 75% precision. Even though it likely had much room for improvement, we abandoned this approach completely. A non-negligible error rate in the first step alone would spoil the chain as the errors in the individual steps would accumulate.

2.2 *Using MST Parser for Clause Recognition*

The experience with the incremental approach clearly indicated that the idea of dividing the recognition task into more steps should be abandoned in favor of a more "holistic" approach that will try to solve the task in a single step. Because the clause recognition is to a certain extent a similar task to full-fledged parsing (the main difference being the size of units we are working with), using a standard parser with adapted data for this task seemed to be a natural solution. We have decided to use the MST parser [7] for this task due to its quality which had been already demonstrated for several languages of various types, Czech and English being among them.

The standard form of data MST parser is the following: the words and punctuation marks (tokens) represent the nodes of a syntactic tree. Each word has four types of values:

1. an original word form
2. a morphological tag
3. a parent node
4. a label of a dependency edge going towards the parent node.

The adaptation of the data for our experiment had been relatively straightforward. Since the whole parse of a sentence always has a tree structure (at least in the FGD formalism, which we adhere to [8]), and since each clause is formed by a subtree of the parse, the clauses themselves must evidently also form a dependency tree. Now, if we see the relation of a word belonging to a clause as a special case of dependency, then the whole output structure (i.e. the three points listed above) can be seen as one dependency tree.

We need the nodes of the tree to represent the clauses on one hand and the tokens of the clauses on the other hand. We'll use the tokens of the sentence to represent both things. So the set of tokens will be the set of nodes of the tree. Each token will represent itself, and some tokens will also represent the clause they belong to. The choice of which token shall represent a clause is clear as each clause has, according to our definition, a *head* among its tokens.

In the tree, we will distinguish between simple tokens and heads of clauses with dependency types: (1) *token dependency* and (2) *clause dependency*. A word that has a clause dependency to its parent represents a clause. Other words can have token dependencies to such a word, thus representing that the dependant is included in the clause represented by

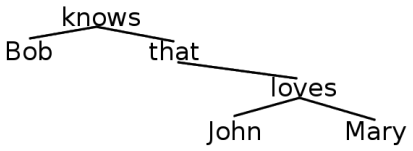

Fig. 1. Standard dependency tree, simplified for illustrating our point.

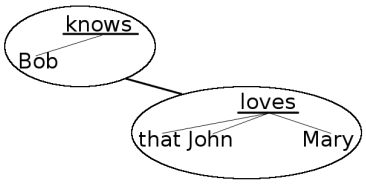

Fig. 2. Dependency tree adapted to capture clausal dependencies. Thin lines denote token dependencies, thick lines denote clause dependencies. Underlined words denote clause heads.

the parent word. Figures 1 and 2 illustrate the difference between a standard syntax tree and an adapted tree that captures clausal relationships on simple made-up English examples.

Krůza, Kuboň 2009 elaborated on details of representing clause structure. Two major factors make the situation more complicated than outlined above: coordination and clauses sharing words. A brief recapitulation follows.

A dependency relationship of one clause to another is represented by clause dependency of the head either to the head of the parent clause or to a word included in the parent clause. The former variant is simpler but the latter can capture which token of the parent clause is the parent of the head of the child clause in the parse. This information does not strictly belong to the clause relationships, where clauses are seen as depending on each other, not on tokens, but choosing to preserve this piece of information in the clause tree is an option that can save a lot of trouble in later stages. Figures 3 and 4 illustrate the difference between the two alternatives.

Beside dependency relations between clauses, two clauses can also be coordinated. This frequent phenomenon, which is in fact a real nightmare for the dependency paradigm, is represented by a special depen-

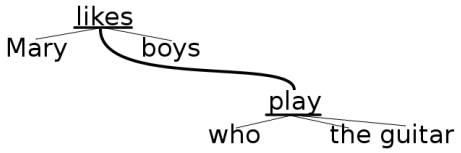

Fig. 3. Illustration of the variant where the dependency relationship of two clauses is delegated on the two heads. The alternative is simpler and seems cleaner in the sense of keeping the clauses atomic.

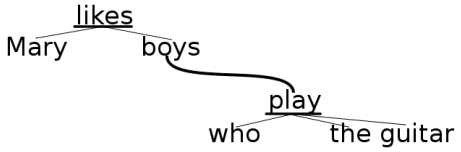

Fig. 4. Illustration of the variant where the dependency relationship of two clauses is delegated on the head of the dependent clause and its actual antecedant from the parent clause. The alternative introduces heterogeny but can be very useful in the stage of reconstructing the actual sentence parse from inter-clausal and intra-clausal trees.

dency type of both coordinated clauses on the coordination. Thus, we introduce a new type of tree nodes. Now, beside clauses and tokens, clause coordinations also occur. They are denoted by having the *coordination dependency*. The coordinated clauses are denoted by having the *member dependency* type.

Table 1 shows an example of the data format for MST parser.

Often, a noun phrase or another constituent forms a modifier shared by both coordinated clauses. Such subtrees are represented by a node of the tree with a special *part dependency* type. A tree with all dependency types is shown in Figure 5. Notice that this short example with all types of dependencies is quite extreme. Typically, a clause-structure tree is much flatter, which is one of the differences compared to more complex trees resulting from full-fledged parsing.

Having such trees, we trained the MST parser to recognize them. The best results were achieved with a second-order model and non-projective algorithm.

Table 1. A sample sentence with formating for the MST parser: "Šetřete peníze, netelefonujte, faxujte! Je tento reklamní slogan pravdivý? [Save money, don't call, fax! Is this advertising slogan true?]"

```
Šetřit         peníze ,           telefonovat_:T
i-P-           NIP4   :---        i-P-
Clause-MEMBER  Word   Coord.-CHILD Clause-MEMBER
3              1      0           5

,              faxovat_:T    !
:---           i-P-          :---
Coord.-MEMBER  Clause-MEMBER Word
3              5             0

být           tento reklamní slogan pravdivý ?
B-S-          DYS1  AIS1     NIS1   AIS1     :---
Clause-Child  Word  Word     Word   Word     Word
0             1      1       1      1        0
```

2.3 *Baseline*

Since our work on this type of automated clause recognition in Czech was pioneer, we had no direct comparison with other approaches. Our task is simpler than full-fledged parsing and it is our ultimate goal to improve automated parsing. So deriving the clauses from the result of an existing parser offers a fitting baseline for us.

We used the development test data from CoNLL 2009, which contains MST-predicted heads and labels. The clauses have been derived using the algorithm presented in [6].

3 EVALUATION

To use the standard precision and recall scores to evaluate clause recognition, we need to define what makes for a correctly-recognized clause. We use two criteria:

1. whether the head is correctly recognized, and
2. whether the component, i.e. set of words of the clause is correctly assigned.

In addition, we also use a conjunction of both criteria. We call these the scores *a head-based score*, *a component-based score* and *a head-and-component-based score*.

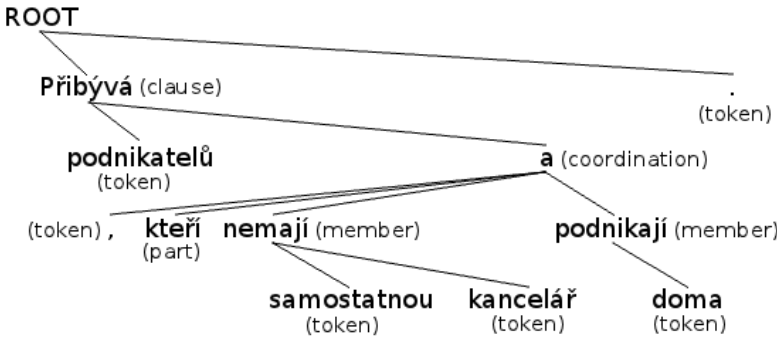

Fig. 5. Clause-structure tree as fed to MST parser. The-number-grows of-businessmen, who don't-have separate office and work at-home. (The number of businessmen who don't have a separate office and work at home, grows.)

None of these, however, gives a continuous measure of the word distribution. We want a way to measure that a clause that has 12 out of 15 words correctly assigned is better than one that has only 9 of 15. For this purpose, we introduce a so-called *fuzzy score*. It is defined by the following algorithm:

1. Initialize an empty precision score array.
2. Initialize an empty recall score array.
3. For each predicted clause P find a clause G in gold standard such that no other clause of gold standard contains more tokens belonging to P than G does.
   Push the evaluation of the following formula to the precision score array: ((#words that P and G share – #words present in P but absent in G) / #words in P)
   End for.
4. For each clause in gold standard G find a predicted clause P such that no other predicted clause contains more tokens belonging to G than P does.
   Push the evaluation of the following formula to the recall score array: ((#words that G and P share – #words present in G but absent in P) / #words in G)
   End for.
5. Return mean values of the precision and recall score arrays.

The idea behind the fuzzy score is taken from precision and recall measures. Precision constituent is simulated by looking at each found

Table 2. Evaluation of the MST-based model on dtest data

Method	Precision	Recall
fuzzy	0.95	0.95
head	0.94	0.94
component	0.78	0.78
head+comp	0.78	0.78

Table 3. Baseline results

Method	Precision	Recall
fuzzy	0.87	0.90
head	0.89	0.86
component	0.59	0.57
head+comp	0.58	0.57

clause, finding the gold clause sharing the most tokens, and calculating the overlap ratio. Analogically, recall is simulated by looking at each gold clause and comparing it to an adequate predicted clause. Since we're comparing two coverages of a given set (a sentence), the score cannot reach zero. The worst case for the recall constituent would be identifying each token as a clause, whereas the worst case for the precision constituent would be identifying the whole sentence as one big clause.

The fuzzy score does its job in measuring component overlap but ignores everything else, so we use it in addition to the aforementioned measures. Table 2 shows the evaluation of the MST-based model. Table 3 shows the evaluation of the baseline.

3.1 *Improving the MST-based Model*

Seeing how much worse the model does at assigning words to clauses in comparison to identifying the clauses themselves and their heads, we were looking into ways of improving the assignment of words into correct clauses. When using the MST in projective mode, the most errors had been done where there was a discontinuity in a clause, e.g. when a clause spanned words 1 to 3 and 6 to 9. Switching to the non-projective algorithm raised the score a touch but there was no observable tendency in the remaining errors any more. We have therefore made an attempt to employ machine learning again and we have trained another model specifically for distributing the words into the pre-identified clauses. This

new model was applied on top of the MST-generated clause structure, so we could use the original MST prediction as a feature.

This new *component model* had couples of words as observation samples. Both words of such couples always belonged to the same sentence; one of them was a head of a clause and the other one was not (in the MST prediction). The predicted feature was whether the first word represents the clause that the other word belongs to. We have defined a set of 148 morphological, lexical and MST-prediction-based features, and planned to use a statistical feature-selection method to get the optimal feature space.

However, we have quickly encountered technical limitations: the training data set was simply too large to fit into the memory. All attempts to train a model failed, either not finishing even after ten days (e.g. k-nearest-neighbor) or crashed on depleting memory (all linear models including SVM). Finally we have tried the C5.0 decision tree. It finished in mere 16 minutes. The induced decision tree was only employing 38 features, which was a neat (though possibly suboptimal) feature selection.

Applying the component model raised the component-based score by 0.1%. The fuzzy score has been raised by 0.0004%. C5.0 was reporting its error rate at 3.9%, which is not bad and certainly not easy to beat by tweaking the feature set.

Because of the negligible contribution, we decided not to use the component model. The obvious explanation for the small contribution seems to be the strictness of the component score, where a clause is only considered correctly identified, when its set of components is correctly assigned. One token off or extra and the clause is not counted as a success. The decent fuzzy score confirms that the token distribution has actually reached a level where improvements are hard to get.

4 CONCLUSION

The paper presents an experiment with a method for automatical clause detection using a specially-trained MST parser. A custom measure rate has been defined to evaluate the recognition. The method outperforms deriving clauses from full-fledged automated parsing with MST. What remains yet to be seen is whether parsing the detected clauses would yield better results than parsing the sentences in a classical way.

ACKNOWLEDGMENTS This work was supported by the Grant of Czech Science Foundation (GAČR) No. P202/10/1333 and by the Charles Uni-

versity Grant Agency (GAUK), Grant No. 920913. In this work we used language resources developed, stored, and distributed by the LINDAT / CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013). Participation in the conference was supported by Foundation of Vilem Mathesius.

REFERENCES

1. Petkevič, V.: Clause identification based on corpora of contemporary Czech. In: Gramatika a korpus / Grammar & Corpora 2007. (2007)
2. Tjong, E.F., Sang, K., Déjean, H.: Introduction to the CoNLL-2001 shared task: Clause identification. In Daelemans, W., Zajac, R., eds.: Proceedings of CoNLL-2001, Toulouse, France (2001)
3. Stevenson, S., Carreras, X.: Proceedings of CoNLL-2009. Association for Computational Linguistics, Boulder, Colorado (2009)
4. Lopatková, M., Homola, P., Klyueva, N.: Annotation of sentence structure: Capturing the relationship between clauses in Czech sentences. Language Resources and Evaluation (2012) 25–36
5. Bejček, E., Panevová, J., Popelka, J., Straňák, P., Ševčíková, M., Štěpánek, J., Žabokrtský, Z.: Prague Dependency Treebank 2.5 – a revisited version of PDT 2.0. In: Proceedings of Coling 2012. (2012) 231–246
6. Krůza, O., Kuboň, V.: Automatic extraction of clause relationships from a treebank. In: Computational Linguistics and Intelligent Text Processing (CICLing 2009). Number 5449 in LNCS. Springer (2009) 195–206
7. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms. In: HLT/EMNLP'05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Morristown, NJ, USA, Association for Computational Linguistics (2005) 523–530
8. Sgall, P.: Generativní popis jazyka a česká deklinace. Academia, Prague, Czech Republic (1967)

OLDŘICH KRŮZA
INSTITUTE OF FORMAL AND APPLIED LINGUISTICS,
FACULTY OF MATHEMATICS AND PHYSICS,
CHARLES UNIVERSITY IN PRAGUE,
MALOSTRANSKÉ NÁMĚSTÍ 25, PRAGUE, CZECH REPUBLIC
E-MAIL: <KRUZA@UFAL.MFF.CUNI.CZ>

VLADISLAV KUBOŇ
INSTITUTE OF FORMAL AND APPLIED LINGUISTICS,
FACULTY OF MATHEMATICS AND PHYSICS,
CHARLES UNIVERSITY IN PRAGUE,
MALOSTRANSKÉ NÁMĚSTÍ 25, PRAGUE, CZECH REPUBLIC
E-MAIL: <VK@UFAL.MFF.CUNI.CZ>