Syntactic Analysis Using Finite Patterns: A New Parsing System for Czech

Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček

Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xkovar3,hales,xjakub}@fi.muni.cz

Abstract. Syntactic analysis of natural languages is considered to be one of the basic steps to advanced natural language processing, such as logical analysis or information retrieval with natural language texts. The Czech language can be characterized as a morphologically rich language with a relatively free word order, which further complicates the problem of syntactic analysis. Current parsing systems for Czech fight many problems including low precision or high ambiguity of the parser output. In this paper, we show a new approach to syntactic analysis of free-word-order languages based on the idea of pattern matching linking rules. The system, named SET, is currently developed and tested with the Czech language as a representative of free-word-order languages with very rich morphological system. We briefly mention current approaches and parsing systems for Czech. Then we describe the basic ideas as well as details of SET's prototype implementation of the pattern matching approach to syntactic analysis. We also offer preliminary analysis of the system parsing precision and discuss the advantages and disadvantages of this approach.

1 Introduction

The goal of the natural language syntactic analysis is to reveal the surface structure of the input sentence. It can be viewed as a "corner stone" of any complex natural language processing (NLP) tasks, ranging from intelligent searching in the text to question answering systems and complex information analysis of the input text.

In the current project, we are concentrating to the syntactic analysis of free-word-order languages. Czech as a representative of free-word-order languages with rich morphology poses barriers to parsing using formalisms that are relatively successful in combination with analytical languages such as English or German [1]. Due to the character of the language, mainly its unrestricted word order, current Czech parsers face problems such as high ambiguity of the analysis output (as a result of high number of rules describing mostly marginal phenomena), or low precision or coverage on corpus texts (as a result of trying to avoid high ambiguity) [2]. At the time, the solution to these problems does not seem to be straightforward.

Z. Vetulani (Ed.): LTC 2009, LNAI 6562, pp. 161–171, 2011.

Our intention in this paper is to show an alternative approach to syntactic analysis, namely for free-word-order languages, and propose another possible solution to the problems outlined above. We describe a system for syntactic analysis of Czech, named SET, based on the principle of pattern matching [3] linking rules, that is not far from techniques used with partial parsing [4]. However, we use these principles to obtain a complete syntactic analysis in the form of parsing trees.

In the next section, we give a quick overview of current approaches in the Czech language parsing; then we continue with the description of the new approach, including motivations, main ideas and a prototype system implementation, called SET. In the last sections, we show the current results of the SET system and conclude with the discussion of advantages and disadvantages of this approach.

2 Current Approaches to Czech Language Parsing

There are two main approaches to the automatic syntactic analysis of Czech at the moment. The first uses the formalism of Functional Generative Description, FGD [5,6], for syntax description and is developed at the Institute of Formal and Applied Linguistics in Prague. Within this formalism, the syntactic information is encoded as an acyclic connected graph of dependency relations, called *dependency tree*.

Based on the FGD formalism, a large manually annotated corpus has been created, the Prague Dependency Treebank – PDT [7]. With its 1.5 million syntactically annotated tokens, the PDT in version 2.0 forms the biggest source of Czech syntactic data. Dependency parsers, e.g. [8,9], usually use the PDT data for training their internal syntax models.

The second approach to Czech language parsing uses the constituent formalism of syntax and its development centre is located at the Natural Language Processing Centre of Masaryk University in Brno. The constituent formalism encodes the syntactic information as a derivation tree based on the formal grammar of the language, in our case Czech. At the NLP Centre, the `synt` parser has been developed. It is based on the metagrammar formalism with a context-free backbone and contextual actions and an efficient variant of the chart parsing algorithm [10].

In both cases, the best results currently achieve 85–87 % precision, with respect to the testing data and precision metric used in the evaluation [11,12,13].

3 SET – Syntactic Analysis as Pattern Matching Linking Rules

Pattern matching belongs to techniques that are widely used in natural language processing [3,14]. The main advantage of pattern matching is transparency and simplicity. Applications based on pattern matching are usually very fast, understandable for people and suitable for further extensions.

In the context of syntax processing, pattern matching is often used in partial parsing and morphological disambiguation [15]. Very interesting and successful commercial application of syntactic patterns are the *word sketches* [16], also available for Czech [17].

We have adapted the notion of a *pattern* to describe probabilistic linking rules that are able to cope with the unrestricted word order of the language. Let us have a finite number of known syntactic constructions – patterns. Then, we take the process of analysis as searching patterns in the input text, sorting them and finally selecting the most probable ones.

As a *pattern*, we take a sequence of word classes that forms a syntactic phenomenon. The simplest example is an adjective followed by a noun, in Czech with case-number-gender agreement, forming a noun phrase (or alternatively, determining an *adjective* → *noun* dependency). The patterns can contain gaps for tokens that are not relevant for the particular phenomenon, e.g. the symbolic pattern *preposition . . . noun* (in Czech with case-agreement) for describing a basic prepositional phrase has a gap for any number of adjectives. More complicated patterns can also be used, e.g. *verb . . . comma sub_conjunction . . . verb* for description of subordinate clause. The realisation of a pattern in the input sentence (the particular words covered by the pattern) is then called a *match*.

In the process of parsing, only a limited number of matches is selected to obtain complete and unambiguous sentence analysis.[1] This selection is driven by a number of factors such as a given probability of particular patterns, the length of the match, its position in the input sentence or values of collocation statistics for words in the match.

3.1 The Parsing Algorithm

Given the set of patterns, the parsing algorithm consists in identifying the patterns in the input and selecting the best matches.

Following one of our main motivations for creating a new parser, namely the simplicity, the matching algorithm has been implemented as a regular expressions matcher, providing both expressivity and efficiency in parsing.

The obvious objection that the natural language structure is at least context-free is not in place because we do not address the decision problem – if the particular sentence is grammatical or not – but just try to find most probable analysis provided the sentence is correct. Basically, the parser assumes every input sentence to be grammatical. Therefore, we consider our approach to be sufficient to reveal the syntactic structure of natural language sentences.

The patterns are divided into groups or *layers* according to the nature of the described phenomena – the layers are applied successively. Currently, SET contains 6 pattern layers.

[1] Ambiguous sentences could be described with more possibilities of how to select the best matches; but at this stage of development, we do not take them into account. We suppose each sentence has only one correct analysis, as it can be found in the annotated corpus.

```
function parse(segment S):
--------------------------
for layer in layers do begin
   found_matches := {};
      // find all matches
   for rule in layer do begin
        new_matches :=
                 find_matches(rule, S);
        found_matches += new_matches;
   end;
      // select the best of them
   best_matches :=
            select_best_matches(
                 found_matches, level);
      // apply selected matches
   for match in best_matches do begin
        add_relationships(S, match);
   end;
end;
```

Fig. 1. Pseudo-code of the parsing algorithm

The first three layers contain patterns describing interjections, "trusted" clause delimiters (e.g. semicolon) and sentence endings. The fourth layer patterns are used for detection of relative clauses boundaries; the fifth layer is dedicated to coordinations. The patterns in the last layer address all the rest – the structures like noun groups, verb groups etc. that are handled as dependency phenomena (see also below).

At each layer, the best non-conflicting match is selected on the basis of ranking functions that involve the pattern rank (defined manually for each pattern), the match length and its position in the input sentence. Finally, based on the selected patterns, a parsing tree is created.

The parsing algorithm is also outlined in the Figure 1. An example run of the parser is provided in Section 4.

3.2 The Pattern Definitions

The algorithm introduced above needs to be given the description of the language patterns. SET includes a readable and expressive definition format for the patterns to be transparent and easily editable by parser developers. In the following paragraphs, we will outline the main principles of the definition format; for a full reference, see the SET project page at `http://nlp.fi.muni.cz/projects/set`.

Each pattern definition consists of two parts, a *template* and a set of *actions*. If a template is matched during parsing, the corresponding action is executed. The template part defines a sequence of word classes involved in the pattern, including gaps. A word class is described with a word form, lemma and/or a morphological category, obtained from the preceding morphological analysis. Using labeled predefined restrictions, e.g. *verb_infinitive*, is also possible.

The actions instruct the parser how to interpret the particular match, e.g. they can contain instructions for marking a dependency relation between two words. They can also specify the rank of the pattern and additional match restrictions such as morphological agreements.

```
TMPL: numeral ... noun
        AGREE 0 2 cgn MARK 0 DEP 2
```

In this example the template on the first line specifies a noun phrase consisting of a numeral and a noun, such as *three dogs* or *three beautiful dogs*. In the action part on the second line, a case-gender-number agreement is required between the two tokens and the parser is instructed to mark a numeral → noun dependency in case the pattern is used.

```
TMPL: verb ... $AND ... verb
        MARK 0 2 4 <coord> PROB 500 HEAD 2
        $AND[word]: , a ani nebo
```

The template in the second example describes a coordination of two verbs using one of the Czech conjunctions *a, ani, nebo* or a comma. The actions say that the relevant tokens should be marked as a coordination with the conjunction being the head of this constituent. Also, the rank of this pattern is increased from default 100 to 500.

```
TMPL: noun $...* comma [tag k3yR] $...* verb $...* rbound
        MARK 2 7 <relclause> DEP 0 AGREE 0 3 gn
        $...*[tag not]: k3yR
```

The last rule provides a complex example of the expressivity the rule system poses. It is used for matching relative clauses. The template is matched for a sequence consisting of a noun followed (possibly) by gap (which cannot be a possesive pronoun as specified on the third line), a comma, a possesive pronoun, again a possible gap and the verb followed by an arbitrary gap and a special tag `rbound` marking the end of the segment. Such a relative clause is enclosed into a `<relclause>` constituent and this constituent is made dependent on the corresponding noun while the agreement in gender and number between both of them is verified.

4 The SET Parser

The SET (Syntax in Elements of Text) system is a prototype implementation of the algorithm presented previously. The program is written in Python and currently includes a set of approximately 150 patterns.

Although the number of patterns will need to grow in the future to increase parsers precision, it is and will be significantly lower than the number of rules that would be needed when using the usual grammar approach. This plays an important role because, as we have mentioned, the maintanenance problems are crucial for rule-based systems.

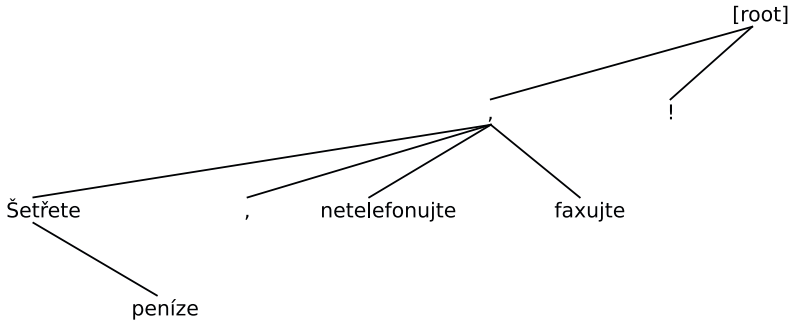

Fig. 2. Example of a usual dependency tree

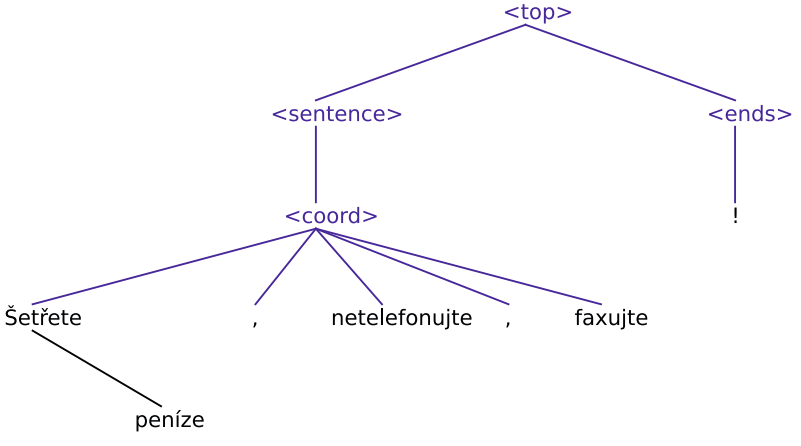

Fig. 3. The corresponding hybrid tree for the same sentence

As its input, the system expects an unambiguously lemmatized and morphologically tagged Czech sentence in a word-per-line format. This simplifies the task – in real applications, we do not have access to correct unambiguous tagging. We can use a tagger that, however, can produce errors and therefore it is not a very good solution. Other possibility is extending the parser so that it is able to handle ambiguous morphological tagging. This extension is straightforward in case of the pattern matching approach and its implementation is in progress.

The parser produces parsing trees in three possible output formats: besides the usual dependency and constituent format, we have introduced a combination of both, a so called *hybrid* format. In this format we profit from the benefits of both the formalisms in particular cases: e.g. we mark coordinations as constituents, as well as numbers, addresses or named entities. In other cases we mostly employ the dependencies to prevent parenthesizing ambiguities that may raise up when using constituents for marking phrases like *nízká rychlost přenosu* (*low speed of*

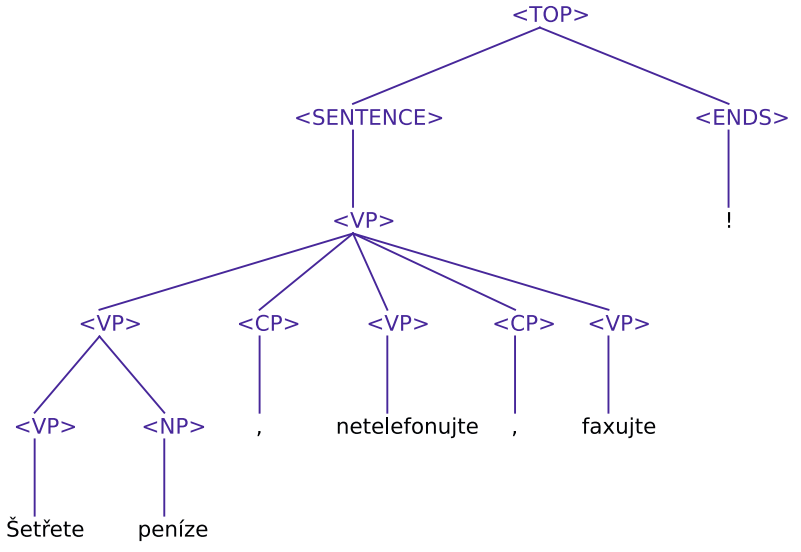

Fig. 4. The corresponding phrasal tree for the same sentence

transmission – in Czech it is not clear whether *speed of transmission* is or is not stronger than *low speed* and this often causes problems).

The difference between a plain dependency tree, hybrid tree and a constituent tree is well-illustrated on the following examples that provide all output forms for the input sentence *Šetřete peníze, netelefonujte, faxujte!*[2]:

Apart from trees, information on all found matches is provided in the output so that ambiguous structures (matches that were found but not selected) can be discovered. An example of the parser run with this additional information follows (still the same sentence, shortened):

```
Parsing segment 0: Šetřete peníze , netelefonujte , faxujte !
--------------
interjections ...
----
ends ...
Match found: !
  Rule: !
Sub-segment created: Šetřete peníze ,
netelefonujte , faxujte
Sub-segment created: !
Phrase created: <top> ::: <sentence> <ends>

Parsing segment 0.0: Šetřete peníze , netelefonujte , faxujte
--------------
interjections ...
```

[2] in English: *Save money, do not phone, fax!*

```
ends ...
hard delimiters ...
relative clauses ...
coordinations ...
Match found: peníze , netelefonujte ,
  Rule: noun $...* comma (...)
Match found: Šetřete , netelefonujte
  Rule: $VERB ... $AND (...)
Match found: Šetřete , faxujte
  Rule: $VERB ... $AND ... $VERB (...)
Match found: Šetřete , faxujte
  Rule: $VERB ... $AND ... $VERB (...)
Match found: netelefonujte , faxujte
  Rule: $VERB ... $AND ... $VERB (...)
Match selected: Šetřete , netelefonujte :: <coord>
Match selected: netelefonujte , faxujte :: <coord>
Phrase created: <coord> ::: Šetřete , netelefonujte , faxujte ::: head = ,
----
dependencies ...
Match found: Šetřete peníze
  Rule: $VERB ... noun MARK 2 DEP 0
Match found: Šetřete peníze
  Rule: $VERB noun MARK 1 DEP 0
Match found: peníze netelefonujte
  Rule: noun ... verb_all MARK 0 DEP 2
Match found: peníze faxujte
  Rule: noun ... verb_all MARK 0 DEP 2
Match selected: Šetřete peníze
Dependency added: Šetřete peníze
```

5 Experiments and Preliminary Results

In this part, we discuss the results of measuring the parser output precision with respect to the PDT data. The results are still preliminary since the set of patterns as well as the ranking functions are still in development, however, they show the possibilities of the pattern matching approach.

In the following, we use the values of the *dependency precision*, i.e. the number of correctly labeled dependencies divided by the number of all dependencies in the tree. In the following, we use the values of the dependency precision, i.e. the number of correctly labeled dependencies divided by the number of all dependencies in the output tree. The corresponding value, *dependency recall*, i.e. the number of correctly labeled dependencies divided by the number of all dependencies in the correct tree, would be in all cases the same (the total number of dependencies in the output tree and in the correct tree is always the same) and therefore it is not included separately.

Also, we do not use *coverage*, i.e. the percentage of sentences accepted by the parser because, as mentioned above, our parser gives an output tree for every sentence. Therefore, the coverage values in all cases would be 100 %.

Table 1. Results – dependency precision

Testing set	Average	Median
PDT e-test	76.14 %	78.26 %
PDT 2000	83.02 %	87.50 %
PDT 50	92.68 %	94.99 %

5.1 Testing Data

We have used three data sets for evaluating the SET parser precision. The first of them is the *PDT e-test* [7] dedicated to testing and comparing dependency parsers. This testing set contains 173,586 words in 10,148 sentences.

Since the PDT data contains very strange sentences such as lists of sport scores or parts of tables [13] (it is debatable whether such constructions should be considered as Czech sentences and SET does not include patterns for their successful parsing), we decided to use the grammar of the `synt` parser [10] to filter these out. According to the latest measurements [18, p. 77], `synt` achieves a very high coverage above 94 % on common Czech sentences, therefore we find it to be safe to use for filtering out nongrammatical texts. As result, the second testing set (*PDT 2000*) consists of 2000 sentences randomly selected from the PDT that were accepted by the grammar of the `synt` parser.

The last testing set (*PDT 50*) consists of 50 sentences randomly selected from the PDT that were used in the process of the SET parser development. These results are not intended to be representative but they can contain some information about the theoretical limits of the approach.

In all testing sets, we have used manually disambiguated morphological tags since at this stage of development, we wanted to measure the precision of the parser only, not the complex analyzers pipeline that can produce errors on lower levels of analysis.

In Table 1, we can see the results for all testing sets. They show that the precision ranges from 76 to 95 percent. The average result for *PDT e-test* is comparable with the third best result on this testing set according to the information available at `http://ufal.mff.cuni.cz/czech-parsing`.[3]

As expected, the *PDT 2000* results are significantly better, which indicates that the parser fails mainly at rare constructions that we have not considered in the design of the patterns so far. Also, higher median values show that the main problems are caused by a small number of uncovered syntactic phenomena rather than by systematic failures. This fact is a motivation to continue in improving the pattern set description.

The 93–95 % precision for the last testing set might not be far from the absolute precision limit that can be reached (due to errors and inconsistencies in corpus annotation). Though the number is not very representative, it may indicate that the pattern matching approach is robust enough to produce better results on bigger data in the future.

6 Conclusions

In the paper, we have introduced a new approach to the syntactic analysis of free-word-order languages based on pattern matching linking rules. We have presented the main ideas, the prototype implementation and the preliminary precision measurements.

[3] With respect to the fact that we have used manually disambiguated morphological tags.

Besides promising precision values, this approach displays several advantages. When compared with the statistical parsers that currently reach the best results for Czech [8,9], the pattern matching approach does not depend on annotated corpus data and the underlying conventions. On the other side, the principle of pattern matching is simple enough to enable future experiments with machine learning from annotated data. Also, the pattern definition format as well as the algorithm of analysis is very transparent, which can significantly speed up the future development.

In the near future, we would like to exploit additional available information in the parsing process, such as verb valencies and corpus collocation statistics. Also, we are going to continue in the development of the pattern set and perform a thorough comparison with other available parsers for Czech.

Acknowledgements

This work has been partly supported by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009 and by the Czech Science Foundation under the project P401/10/0792 and by the EU project PRESEMT (ICT-248307).

References

1. Baumann, S., Brinckmann, C., Hansen-Schirra, C., et al.: The muli project: Annotation and analysis of information structure in german and english. In: Proceedings of the LREC 2004 Conference, Lisboa, Portugal (2004)
2. Horák, A., Kadlec, V., Smrž, P.: Enhancing Best Analysis Selection and Parser Comparison. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2002. LNCS (LNAI), vol. 2448, pp. 461–467. Springer, Heidelberg (2002)
3. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval. ACM Press, New York (1999)
4. Mráková, E., Sedláček, R.: From Czech morphology through partial parsing to disambiguation. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 126–135. Springer, Heidelberg (2003)
5. Sgall, P.: Generativní popis jazyka a česká deklinace (Generative Description of the Language and the Czech Declension). Academia, Prague (1967)
6. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence and Its Semantic and Pragmatic Aspects. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands (1986)
7. Hajič, J.: Complex Corpus Annotation: The Prague Dependency Treebank, Bratislava, Slovakia, Jazykovedný ústav Ľ. Štúra. SAV (2004)
8. Hajič, J., Collins, M., Ramshaw, L., Tillmann, C.: A Statistical Parser for Czech. In: Proceedings ACL 1999, Maryland, USA (1999)
9. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-Projective Dependency Parsing using Spanning Tree Algorithms. In: Proceedings of HTL/EMNLP 2005, Vancouver, BC, Canada (2005)
10. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008)
11. Holan, T., Žabokrtský, Z.: Combining Czech Dependency Parsers. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2006. LNCS (LNAI), vol. 4188, pp. 95–102. Springer, Heidelberg (2006)

12. Kovář, V., Jakubíček, M.: Test suite for the Czech parser synt. In: Proceedings of Recent Advances in Slavonic Natural Language Processing 2008, Brno, Czech Republlic, Masaryk University, pp. 63–70 (2008)
13. Horák, A., Holan, T., Kadlec, V., Kovář, V.: Dependency and Phrasal Parsers of the Czech Language: A Comparison. In: Matoušek, V., Mautner, P. (eds.) TSD 2007. LNCS (LNAI), vol. 4629, pp. 76–84. Springer, Heidelberg (2007)
14. Sojka, P.: Competing patterns for language engineering. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2000. LNCS (LNAI), vol. 1902, pp. 157–162. Springer, Heidelberg (2000)
15. Przepiórkowski, A., Buczyński, A.: ♠: Shallow parsing and disambiguation engine. In: Proceedings of the 3rd Language & Technology Conference, Poznań (2007)
16. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proceedings of the Eleventh EURALEX International Congress, Lorient, France, Universite de Bretagne-Sud, pp. 105–116 (2004)
17. Rychlý, P., Smrž, P.: Manatee, Bonito and Word Sketches for Czech. In: Proceedings of the Second International Conference on Corpus Linguisitcs, pp. 124–132. Saint-Petersburg State University Press, Saint-Petersburg (2004)
18. Kadlec, V.: Syntactic analysis of natural languages based on context-free grammar backbone. PhD thesis, Masaryk University (2008)