Probabilistic Head-Driven Chart Parsing of Czech Sentences

Pavel Smrž and Aleš Horák

Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {smrz,hales}@fi.muni.cz

Abstract. In this paper we present the results of our work on an implementation of a fast head-driven chart parser for Czech language and constructing the appropriate grammar covering all prevailing grammatical phenomena of Czech. We reassume our previous work on syntactic analysis that was based on the GLR mechanism. We have extended our metagrammar formalism so as to reinforce the declarativeness of the linguistic description. With respect to the massive ambiguity of the grammar we have enriched the head-driven chart parsing mechanism with probabilities obtained from training tree-bank corpus.

1 Introduction

There are many grammar formalisms for representing the natural language phenomena that usually use feature structures. The most popular ones are probably HPSG [1], LFG [2] and LTAG [3]. These formalisms are well known for their linguistic adequacy, however, they suffer from problems with high complexity of parsing algorithms. To overcome this difficulty we have introduced the concept of metagrammar [4] that is constructed on a context free grammar backbone which enables us to use a CFG parser modified for the sake of feature agreement satisfaction and other linguistic tests and actions essential for parsing a free word order language like Czech.

Taking into consideration the high ambiguity of free word order languages on the syntactic level one soon comes to the need of a tool for choosing among the huge number of resulting parses. Such tools are often based on probabilities automatically acquired from syntactically tagged training sentences. A stochastic model of context free parsing with the longest tradition is the probabilistic CF grammar [5], which combines the classical CF rules with quantities characterizing how likely the rule applications are. Nevertheless, this mechanism usually needs an extension that captures the context dependency in the derivation process. Our system described below exploits the probabilistic approach for assorting the set of output syntactic structures.

2 Grammar and Parser Design

In this section we describe the insides of our system from the point of the grammar formalism and the actual parser implementation. We also present new combinatoric constructs that substantially reduce the number of verb part rules written by linguists (which form nearly 60% of our grammar).

The only grievance to the output of our parser from the linguistic point of view regarded the complexity of derivation trees, namely the high number of levels in trees. We have therefore provided the possibility to postprocess the output with the aim to specify the importance of selected nonterminals that are then displayed in the result.

2.1 Metagrammar Uplift

The metagrammar consists of a set of CF rules extended with global order constraints, combinatoric constructs and special flags restricting the generative process. The basic metagrammar constructs are `order()`, `rhs()` and `first()` as described in [4].

In the current version we have added two generative constructs and the possibility to define rule templates to simplify the creation and maintenance of the grammar. The first construct is formed by a set of `%list_*` expressions, that automatically produce new rules for a list of the given nonterminals either simply concatenated or separated by comma and co-ordinative conjunctions:

```
/* (nesmim) zapomenout udelat - to forget to do */
%list_nocoord vi_list
vi_list -> VI

%list_nocoord_case_number_gender modif
/* velky cerveny - big red */
modif -> adjp

/* krute a drsne - cruelly and roughly */
%list_coord adv_list
adv_list -> ADV

%list_coord_case_number_gender np
/* krasny pes - beautiful dog */
np -> left_modif np
...
```

The endings `*_case`, `*_number_gender` and `*_case_number_gender` denote the kind of agreement between list constituents. The incorporation of this construct has decreased the number of rules by approximately 15%.

A significant portion of the grammar is made up by the verb group rules. Therefore we have been seeking for an instrument that would catch frequent repetitive constructions in verb groups. The obtained addition is the `%group` keyword illustrated by the following example:

```
%group verb={
    V:head($1,intr)
      add_verb($1),
    VR R:head($1,intr)
         add_verb($1)
 set_R($2)
}

/* ctu - I~am reading */
/* ptam se - I~am asking */
clause ====> order(group(verb),vi_list)
```

Here the group `verb` denotes two sets of non-terminals with the corresponding actions that are then substituted for the expression `group(verb)` on the right hand side of the `clause` non-terminal.

2.2 Large Scale Parsing

The choice of employed grammar formalism does not directly force us to use a particular parsing strategy. Our previous implementation was derived from generalised LR approach (Tomita parsing) [6]. The results we obtained have been excellent for common sentences, nevertheless we could find some examples that were not parsed in a feasible time. That is why we started to implement a head driven chart parser extended by three kinds of unification actions and probabilistic selection from agenda. This work has already been recompensed with admirable time response for all input sentences.

The number of rules may differ significantly in particular grammars. Extreme values are reached in grammars that are obtained automatically from corpus processing [7]. Our parser is designed in order to cope with drastic increases in number of rules without the loss of its speed. We use as an experiment a grammar of about 35000 rules that were expanded from the base rules plus the unification actions and the rise of analysis time is negligible. Even several hundred thousand rules (created by multiplying the ruleset) are no challenge for the analyser.

The chart parsing techniques for extended CFGs are often underspecified with respect to the way how and when the rule constraints are evaluated. An elegant solution is the conversion of the agreement fulfilment actions to the CF rules. For instance, a grammar rule

```
pp -> prep np
  agree_case_and_propagate($1,$2)
```

is transformed to

```
pp1 -> prep1 np1
pp2 -> prep2 np2
pp3 -> prep3 np3
...
```

In Czech, similarly to other Slavic languages, we have 7 grammatical cases (nominative, genitive, dative, accusative, vocative, locative and instrumental), two numbers (singular and plural) and three genders (masculine, feminine and neuter), while masculine exists in two forms — animate and inanimate. Thus, e.g., we get 56 possible variants for a full agreement between two constituents. The expanded grammar form consists of more than 10000 rules in our case.

Since the number of rules that we need to work with is quite big, we need efficient structures to store the parsing process state. For example, the chart parser implementation used in our experiments employs 6 hash structures — two for open edges, two for closed edges, one hash table for the grammar rules (needed in the prediction phase) and one for all edges in agenda or in chart (the hash key is made of all the attributes of an edge — the rule, the dot position and the surface range).

The gain of this rather complex structure is the linear dependency of the analysis speed on the number of edges in the resulting chart. Each edge is taken into considerations two times — when it is inserted into the agenda and when it is inserted into the chart. The overall complexity is therefore $2k$, where k is the number of edges in the resulting chart.

In the next table we present running times and the number of edges in the resulting chart. We have chosen a set of longer sentences (more than 40 words) from a Czech corpus to demonstrate the efficiency of our parser implementation.

Sentence number	**number of words**	**number of edges**	**time (s)**
1074	47	170941	2.88
1100	52	143044	2.53
1236	42	144286	2.77
1654	51	361072	6.81
1672	59	434430	7.13
1707	40	205345	3.44
2079	66	223063	3.75
2116	49	259899	3.91
2284	42	452797	7.55
2300	60	443022	7.28
2306	102	715835	10.95

3 Analysis Guided by Statistical Data

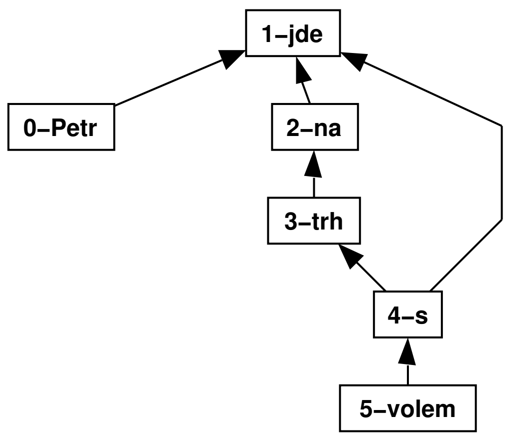
Ambiguity is the fundamental property of natural language. Perhaps the most oppressive case of ambiguity manifests itself on the syntactic level of analysis.

In order to face up to the high number of obtained derivation trees we define a sort order on the output trees that is specified by probabilities computed from appropriate edges in the chart structure. The statistics is also involved in the process of sorting out the edges from agenda in the order that leads directly to n most probable analyses.

A common approach to acquiring the statistical data for analysis of syntax employs learning the values from a fully tagged tree-bank training corpus. Building of such corpora is a tedious and expensive work and it requires a team cooperation of linguists and computer scientists. At present the only source of Czech tree-bank data is the Prague Dependency Tree-Bank (PDTB) [8], which catches dependency analyses of about 20000 Czech sentences.

In order to be able to exploit the data from PDTB, we have supplemented our grammar with the dependency specification for constituents. Thus the output of the analysis can be presented in the form of pure dependency tree. In the same time we unify classes of derivation trees that correspond to one dependency structure. We then define a canonical form of the derivation to select one representative of the class that is used for assigning the edge probabilities.

The dependency structures for all possible analyses are stored in the form of packed dependency graph. An example of the graph for the sentence `Petr jde na trh s volem` (literally: *Peter goes to the market with an ox.*) looks like



The packed dependency graph enables us to recover all the possible standard dependency trees with some additional information gathered during the analysis. The example graph represents two dependency trees only, however, in the case of more complex sentences, especially those with complicated noun phrases, the saving is much higher.

4 Conclusions

The implemented system has already approved that it does not end in its experimental stage. One of many possible exploitations of the syntactic parser is its incorporation into the project of the proposed algorithm for translating natural language sentences into constructions of transparent intensional logic (Normal Translation Algorithm) [9].

The packed dependency graph should also be supplemented with the possibility to impose edge incompatibilities when the incorporated trees contain forbidden combinations of dependencies. This procedure will be based on checking the valency frames of the given verb.

References

1. Pollard, G. and Sag, I.: *Head-Driven Phrase Structure Grammar*, University of Chicago Press, 1994.
2. Neidle, C.: Lexical-Functional Grammar, *Encyclopedia of Language and Linguistics*, New York, Pergamon Press, 1994.
3. Schabes, Y., Abeille, A. and Joshi, A., K.: Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars, In *Proceedings of the 12th COLING*, pp. 578–583, Budapest, Hungary, 1988.
4. Smrž, P. and Horák, A.: Implementation of Efficient and Portable Parser for Czech, In *Proceedings of TSD'99*, pp. 105–108, Springer-Verlag, Berlin, 1999.
5. Manning, C., D. and Schütze, H.: *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, 1999.
6. Tomita, M.: *Efficient Parsing for Natural Languages: A Fast Algorithm for Practical Systems*, Kluwer Academic Publishers, 1986.
7. Moore, R., C.: Improved Left-Corner Chart Parsing for Large Context-Free Grammars, In *Proceedings of the 6th IWPT*, pp. 171–182, Trento, Italy, 2000.
8. Hajič, J.: Building a Syntactically Annotated Corpus: The Prague Dependency Treebank, In *Issues of Valency and Meaning*, pp. 106–132, Karolinum, Prague, 1998.
9. Smrž, P. and Horák, A.: Determining Type of TIL Construction with Verb Valency Analyser, In *Proceedings of SOFSEM'98*, pp. 429–436, Springer-Verlag, Berlin, 1998.