Building Sub-corpora Suitable for Extraction of Lexico-Syntactic Information

Ondřej Bojar
Institute of Formal and Applied Linguistics, ÚFAL MFF UK, Malostranské náměstí 25, CZ-11800 Praha, Czech Republic
obo@cuni.cz

Abstract.

Accuracy of automatic syntactic analysis of natural languages with relatively free word order (such as Czech) can be hardly improved without building large and precise lexicons of syntactic behavior of individual words (e.g. lexicons of verb valency frames). The current treebanks available do not cover enough words. Larger corpora lack the syntactic annotation and many sentences contained in them are too complex to extract the information easily or even automatically. The system AX (automatic extraction) was developed to perform selection of morphologically analyzed sentences by means of custom hand-written rules. The rules can be easily formulated to perform linguistic-motivated filtration. The system AX allows to perform partial syntactic analysis in order to check the occurrence of more complex linguistic phenomena.

1 Motivation

At the current stage of the development, the accuracy of automatic syntactic analyzers of natural languages (in particular Czech) is limited due to the lack of large and precise lexicons of syntactic behavior of individual words (verb valency frames are the most important example). Building such lexicons by hand is rather a time-consuming task and any kind of automatic preprocessing would help.

The available data sources include corpora annotated on different levels of linguistic description. Syntactic information can be easily extracted from corpora annotated on the syntactic level (such as the Prague Dependency Treebank, PDT[1], Böhmová et al. (2001)) and some attempts to extract for example verb subcategorization frames from treebanks were already performed[2]. However the number of occurrences of individual lexical items in

[1] `http://ufal.mff.cuni.cz/pdt/`
[2] Such as (Sarkar and Zeman, 2000).

Proceedings of the Eighth ESSLLI Student Session
Balder ten Cate (editor)

such corpora is usually not sufficient. Bojar (2002) shows that only a few hundreds of verbs have enough occurrences in PDT and that more than 60% of 26,000 verbs found in the Czech National Corpus (CNC[3]) are not covered in PDT at all. Therefore, it is necessary to extract the lexico-syntactic information from larger corpora (such as the CNC) or any texts available.

However, not all sentences containing a given lexeme can serve as a good example to extract the syntactic information (for instance, if two verbs are present in one clause, their complements and adjuncts can be arbitrarily intermixed). And moreover, as the syntactic annotation in these corpora is missing, the sentences can often be too complex to be analyzed by any of the available parsers at a reasonable level of accuracy.

I developed the system AX to simplify the task of selecting feasible examples of sentences for extracting a specific lexico-syntactic information (not just the verb valency frames). The sentences can be easily selected on linguistically based criteria. Partial syntactic analysis of sentences is possible, in order to be able to answer more complex linguistic questions about the sentence.

In section 2 I describe the overall architecture of the system AX. Section 3 gives a brief description of the basic data structure, variant feature structure. Sections 4 and 5 describe the core of the scripting language AX: filters to reject sentences and rules to perform (partial) syntactic analysis. In the last section, I illustrate the use of the system to select sentences suitable for extracting valency frames of Czech verbs and document the improvement of accuracy of Czech parsers when applied only to the selected sentences.

2 The Architecture of AX

The system AX combines the idea of regular expressions and replacements (see (Karttunen et al., 1996; Aït-Mokhtar and Chanod, 1997) and others) with the idea of feature structures (see below) known from unification-based parsers. This combination leads to a formalism that is both, strong to describe complex linguistic properties of sentences of natural language and efficient in the process of parsing.

The user prepares a script of filters and rules to select sentences suitable for a specific purpose. The system AX loads the script and then expects sentences augmented with their morphological annotation (in format of the PDT[4]) on the standard input. The input sentences may or may be not morphologically disambiguated. For every input sentence, the system runs the script and checks, if the sentence passed all the filters or has been rejected. For sentences that pass (referred to with the term "selected sentences"), the output of the final phase (see below) is printed out. This output is for some

[3] `http://ucnk.ff.cuni.cz/`

[4] See `http://shadow.ms.mff.cuni.cz/pdt/Corpora/PDT_1.0/Doc/morph.html`

purposes already suitable for collecting the lexico-syntactic information so that no other parser to process the sentences is needed.

In the following, I describe the overall running scheme of AX. The input sentence is internally stored as a sequence of feature structures that correspond one to one to input word forms. (See section 3 for details.) The input sentence is then processed through a pipe of consecutive *blocks (phases) of operation*. Each of the blocks is either a *filter*, or a *set of rules*.

The input for each block is a set of sequences of feature structures (referred to with the term *the set of "input readings" of the sentence*). If the block is a filter, it checks all the input readings and possibly rejects some of them. If the block is a set of rules, it updates every input reading with all applicable rules and returns a larger set of new readings (it "generates" new readings).

Consecutive blocks are connected, so that the output set of readings from the former block is used as the input set of readings for the latter block. The first of the blocks receives as input the input sentence, the output from the last block is printed out. The order and type of the blocks is up to the author of the script. All the input sentences that were not rejected by any of the filters are *accepted* and the output produced for each of them can also be used to extract the lexico-syntactic information, if appropriate.

A sample flow of readings is demonstrated in figure 1.1.

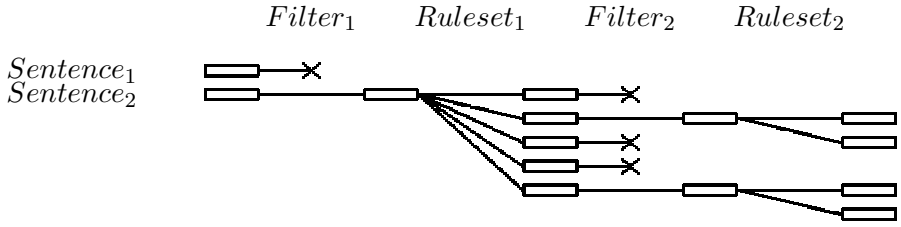


Figure 1.1: Progress of sentences through an AX script. The first input sentence was rejected by the first filter. The second sentence passed the filter and several readings were obtained by the rules in ruleset 1. Some of the readings were then rejected by filter 2 and some passed. Altogether four different readings were then produced by the last ruleset.

3 Feature Structures with Variants

Feature structures (also called attribute-value matrices) allow representing of various linguistic information in a compact and natural way.[5]. For the purposes of this work, untyped feature structures with alternatives (variants) of values are sufficient and serve well to represent very rich and often very

[5]For a detailed characteristic of typed feature structures see Penn (2000).

ambiguous morphological information[6] as well as arbitrary user flags useful in the process of filtering and generating new readings by rules.

The basic operation with two variant feature structures is *unification*. The output of the unification is a feature structure that holds information from both the input structures.[7] For instance:

$$\begin{bmatrix} \text{name} & \text{Kamil} \\ \text{surname} & \{\text{Horak, Klement}\} \end{bmatrix} \text{ and } \begin{bmatrix} \text{surname} & \text{Horak} \\ \text{age} & \text{int(32)} \end{bmatrix} \quad \text{unify and the result is} \quad \begin{bmatrix} \text{name} & \text{Kamil} \\ \text{surname} & \text{Horak} \\ \text{age} & \text{int(32)} \end{bmatrix}$$

Unification fails, if both the features contain an attribute of the same name but a non-unifying value.

For every input word in a sentence, the morphological analysis gives all possible lemmas and morphological attributes of the given word form. This ambiguous morphological information can be stored in a single feature structure with variants. See figure 1.2 on the facing page for an example. The whole sentence of word forms can therefore be stored as a list of feature structures of the same length.

4 Filters

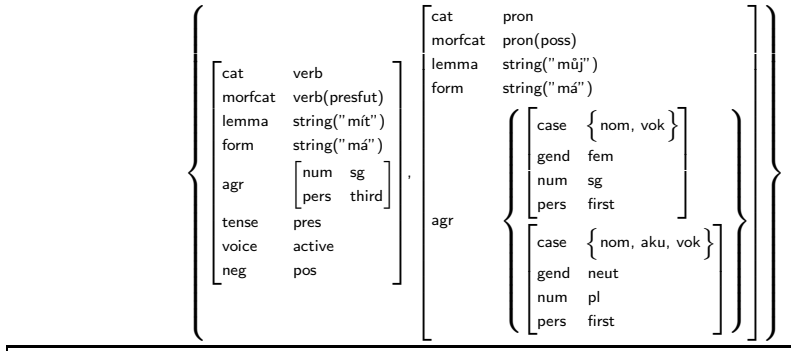
Filters in the language AX are expressed in the form of *regular expressions of feature structures*. The basic differences between common regular expressions (used for instance in many Unix tools) and regular expressions of feature structures used in the language AX are:

- The primitive element of regular expressions is no longer a character, but rather a feature structure. In scripting language AX, the feature structure can be expressed either explicitly or by a shortcut name[8].
- When searching for a subsequence of feature structures that matches a given regular expression, the system checks whether the input structure unifies with the structure in the expression. (Rather than checking two characters for equality.)

[6] In Czech, approx. 4,000 different tags are defined, half of which actually occurred in the Czech National Corpus. For many word forms several dozens of morphological tags are possible (7 different cases · 4 genders · 2 (sg/pl) = 56 possible tags).

[7] If more variants of a value are available, the output will carry out the intersection (more precisely the product of unification of all possible combinations of input variants).

[8] For example, specific types of pronouns can syntactically serve as nouns or adjectives. It is useful to define a shortcut of a feature structure that would match a noun or a noun-like type of pronoun etc. In filters and rules, it is then possible to introduce the whole structure only by its shortcut name.




```
[
  cat - verb, morfcat-verb(presfut), lemma - "mít", form - "má",
  agr - [ num-sg, pers-third ], tense - pres, voice - active, neg-pos
|
  cat - pron, morfcat - pron(poss), lemma - "můj", form - "má",
  agr - [ case - nom;vok, gend - fem, num - sg, pers - first
        | case - nom; aku; vok, gend - neut, num - pl, pers - first ]
]
```

Figure 1.2: This feature structure represents all the possible morphological analyses of the word form *má* which can be a word form of two different lemmas ($mít_{\texttt{verb}}$ and $můj_{\texttt{pronoun}}$ in different cases, numbers and genders). Below is the same feature structure expressed in syntax of the scripting language of AX.

Details of the syntax of the filters are described in Bojar (2002), here I give just a brief example of two different filters:

```
filter reject_more_than_two_verbs:
  .* verb .* verb .*
end

keep "Keep only sentences with exactly one verb
      or those not containing any conjunction":
  !verb* verb !verb* | !conj*
end
```

The keyword `filter` means: reject the sentence if it (as a whole) matches the given regular expression. The meaning of the keyword `keep` is: reject the sentence if it doesn't match the given expression.

5 Rules

Rules are used to modify the input readings of a sentence and generate new readings. Rules have always this form:

`rule` <rule name> :
<replacement> `--->` <input regular expression> ::
<constraints>
`end`

The rule is applied as follows:

- The input sequence of feature structures is searched in order to find a subsequence that matches the <input regular expression> and the <constraints>.
- The obtained subsequence of feature structures is replaced with the <replacement>.

By default, the input sequence is searched for all possible subsequences matching the regular expression and constraints, therefore the rule can produce for one input reading several output readings. In many situations, this nondeterministic approach leads to more output readings than the user actually wants. For such cases, the user can write special keywords in the arrow in the rule to make the rule substitute for instance only the first matching subsequence of feature structures.

By default, the output of one rule within a ruleset is used as input for another rule in the same ruleset (possibly reusing the rule itself). All possible combinations of applying rules are attempted and all the possible outcomes are collected to build the final output set of possible readings for this block (phase) of operation. This nondeterministic behavior can be restricted in several ways: rules can be limited in number of allowed applications[9], an output from one rule can be included in the final set only if no other rule was able to change it, and others. A detailed description of all the options is out of the scope of this paper.

In order to "compute" the <replacement> from the subsequence found in the input, one can use *variables*. The variables can be used in all parts of rules: from the <input regular expression> they get their initial value. The value is then restricted or updated by the <constraints> and their final value is given to the output in the <replacement>.

All the variables can hold a feature structure. The scope of the variables is limited for one application of one rule, that is all the variables are local for the rule and within one application.

The <constraints> are expressed as an unordered list of requirements on variables values. All the requirements must be fulfilled for the rule to be applicable. The constraints can only require certain feature (sub)structures to

[9]In fact, the system AX will not start unless all non-shortening rules, i.e. the rules that are able to produce output not shorter than the input reading, have this maximum number of applications explicitly expressed as a function of the number of input structures. This effectively blocks out the possibility to loop ad infinitum.

unify. By means of these requirements, output variables also get their value. The following example shows a rule to perform a reduction: it combines an adjective and a noun together:

```
rule out_noun ---> adj noun ::
  adj.agr = noun.agr,
  out_noun = noun
end
```

The constraint `adj.agr = noun.agr` guarantees the congruence of the noun and the adjective in case, gender and number. The constraint `out_noun = noun` initializes the output variable with the feature structure of the noun *after* it was already restricted in case, number and gender due to the congruence requirement. In this way, the input morphological ambiguity is step by step solved.

The output <replacement> may copy parts of the input subsequence. This allows an easy formulation of rules that combine distant feature structures in the input sentence. The regular expression can then be used to restrict what can stay between the two (or more) feature structures. As a nice example I show the rule that combines two parts of a Czech verb – the auxiliary part ($být_{tobe}$, which can have several forms, such as $jsem_{\texttt{I}am}$) and the main verb (such as *zalít*, which also can have several forms). The rule also checks the parts for congruence[10]:

```
rule complex_past_tense:
  complex \gap trace
  ---->
     zalil {gap:!{verb,comma,conj}*} jsem
   | jsem {gap:!{verb, comma, conj}*} zalil
  ::
  zalil <- morfcat, voice -> ‘zalil‘,
  zalil = [cat-verb],
  jsem.cat = ‘jsem‘.cat,
  jsem <- morfcat, lemma, neg -> ‘jsem‘,
  jsem.agr = [pers-first;second],
  zalil <- agr.num, agr.gend -> jsem,
  complex = [cat-complexpast],
  complex <- lemma, form, neg, morfcat -> zalil,
  complex <- agr.pers, agr.num, agr.gend, mood -> jsem
  trace = [cat-trace, form-"XtraceX"]
end
```

[10]The scripting language AX has a shorter form of expressing several unification requirements on two variables at once. The constraint: “`usnul <- cat, agr.num, agr.gend -> jsem`” is equivalent with these three: “`usnul.cat = jsem.cat, usnul.agr.num = jsem.agr.num, usnul.agr.gend = jsem.agr.gend`”

The rules find such a subsequence of feature structures that begins with the auxiliary verb and ends with the main verb (or vice versa). The gap between the parts of verb must not contain any other verb, comma or conjunction (introduced by means of shortcuts, see above). The gap gets a label "`gap`", so that it can be copied to the output replacement. As the output, the rule produces a single feature structure representing the complex verb followed a copy of the `gap` section and an auxiliary trace at the place where the other part of the verb was found. Naturally, the trace can be omitted if not needed by any other rules of filters.

6 A Sample AX Usage and Results

The system AX was used to select sentences suitable for extracting typical complements of verbs, namely verb valency frames. The script for this purpose contained 15 filters and 21 rules and selected sentences where the complements of verbs would be easy to observe and excluded sentences that are too difficult to parse. Approximately 15 to 20% of sentences from the Czech National Corpus are selected by this script. I call them "very simple sentences".[11]

So far, the task of actually extracting verb valency frames from the selected sentences was not performed, neither manually nor automatically.[12] Anyway, the utility of the described sentence preselection can be illustrated by measuring the improvement of accuracy of parsers available for Czech (Collins et al. (1999), Zeman (1997, 2002) and a parser by Zdeněk Žabokrtský (unpublished)). All the parsers were tested on all the sentences in the evaluation part of the Prague Dependency Treebank and separately on the selected "very simple sentences" only.

The results show that the best parser available for Czech, the Collins parser, is able to correctly observe 55% of verb frames[13]. When used on very simple sentences only, this measure increases by 10%. A similar result can be achieved by using the parser on short sentences only: in sentences with at most ten words, the Collins parser correctly observes 68% of verb frames. A combination of both filters, short and "very simple" sentences, is however still 5% better, exceeding 73% of correctly observed verb frames.

[11] Optionally, sentences containing "suspicious word order patterns" (WOP) were also rejected. Straňáková-Lopatková (2001) cautiously analyzes the risks of syntactic ambiguity of noun and prepositional phrases in Czech. She defines those WOPs, where there it is not possible to decide whether a noun phrase depends on another noun phrase or on the verb itself. These examples would spoil the observation of verb frames, but it is easy to filter them out using the AX. Full description of the linguistically motivated filters exceeds the scope of this paper, see (Bojar, 2002) for details.

[12] In (Bojar, 2002) I propose an algorithm to extract surface verb frames and describe several open problems of the task of inferring verb valency frames from the surface ones.

[13] That is to correctly identify all immediate daughters of a verb in the dependency tree and not mark any extra nodes as daughters of the verb.

Also quite interesting is the improvement of the traditional accuracy measure, namely the number of correctly assigned dependencies. If used on very simple sentences, the parsers achieve an accuracy of 5 to 10 percent better, up to 88% for Collins. On short and "very simple" sentences, 91.4% for Collins is achieved.

7 Conclusions

In this paper I described a system to perform selection of sentences from corpora based on linguistically motivated criteria. The system allows an easy formulation of filters and also rules for partial syntactic analysis of sentences, if needed to check for more complex phenomena. The sentences can be selected for many purposes: as an automated preprocessing to supply lexicographers with more relevant examples of sentences as well as a first step in a fully automatic extraction of lexico-syntactic information. If the script of filters and rules is cautiously designed, already the output produced from the system AX can serve as raw input to build a lexicon. The expressive power of filters and rules of AX is better than any corpus searching tool known to me. The system was used to select sentences suitable to extract verb valency frames and the utility of this selection was illustrated by improvement of three parsers' accuracy.

In future work, I will use the described system in a large scale to extract valency frames of verbs from the whole Czech National Corpus. I will also try to design an AX script to select sentences suitable for extraction of valency frames of nouns.

8 Acknowledgments

The scripting language presented in this article was developed as a part of my master thesis, Bojar (2002). I would like thank to the supervisor of the thesis, RNDr. Vladislav Kuboň, Ph.D., as well as to all the researchers and staff at the Center for Computational Linguistics, Charles University, Prague. This work was partially supported by the grant GAČR 201/02/1456 and GAUK 300/2002/A INF-MFF.

References

Aït-Mokhtar, Salah and Jean-Pierre Chanod. 1997. Incremental Finite-State Parsing. In *Proceedings of ANLP'97*, pages 72–79, Washington, March 31st to April 3rd.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The Prague Dependency Treebank: Three-Level Annotation Scenario.

In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Bojar, Ondřej. 2002. Automatická extrakce lexikálně-syntaktických údajů z korpusu (Automatic extraction of lexico-syntactic information from corpora). Master's thesis, ÚFAL, MFF UK, Prague, Czech Republic. In Czech.

Collins, Michael, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*, pages 505–512, University of Maryland, College Park, USA.

Karttunen, Lauri, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. 1996. Regular Expressions for Language Engineering. *Natural Language Engineering*, 2(4):305–328.

Penn, Gerald. 2000. *The Algebraic Structure of Attributed Type Signatures*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

Sarkar, Anoop and Daniel Zeman. 2000. Automatic Extraction of Subcategorization Frames for Czech. In *Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000)*, Saarbrücken, Germany. Universität des Saarlandes.

Straňáková-Lopatková, Markéta. 2001. Homonymie předložkových skupin v češtině a možnost jejich automatického zpracování (Ambiguity of prepositional phrases in Czech and possibilities for automatic treatment). Technical Report TR-2001-11, ÚFAL/CKL, Prague, Czech Republic. In Czech.

Zeman, Daniel. 1997. A Statistical Parser of Czech. Master's thesis, ÚFAL, MFF UK, Prague, Czech Republic. In Czech.

Zeman, Daniel. 2002. Can Subcategorization Help a Statistical Parser? In *Proceedings of the 19th International Conference on Computational Linguistics (Coling 2002)*, Taibei, Tchaj-wan. Zhongyang Yanjiuyuan (Academia Sinica).