Techniques for Accelerating a Grammar-Checker

Karel Oliva
Computational Linguistics
University of Saarland
Postfach 15 11 50
D - 66 041 Saarbrücken
Germany
email: oliva@coli.uni-sb.de

Abstract

The paper describes several possibilities of using finite-state automata as means for speeding up the performance of a grammar-and-parsing-based (as opposed to pattern-matching-based) grammar-checker able to detect errors from a predefined set. The ideas contained have been successfully implemented in a grammar-checker for Czech, a free-word-order language from the Slavic group.

1 Introduction

This paper describes an efficiency-supporting tool for one of the two grammar-checker technologies developed in the framework of the PECO2824 Joint Research Project sponsored by the European Union. The project, covering Bulgarian and Czech, two free-word-order languages from the Slavic family, was performed between January 1993 and mid 1996 by a consortium consisting of both academic and industrial partners.

The basic philosophy of the technology discussed in this paper[1] is that of linguistic-theoretically sound grammar-and-parsing-based machinery able to detect, by constraint relaxation, errors from a predefined set (as opposed to pattern-matching approaches, which do not seem promising for a free word-order language). The core of the system (broad-coverage HPSG-based grammars of Bulgarian and Czech, and a single language-independent parser) was developed in the first three years of the project and was then passed to the industrial partners Bulgarian Business System IMC Sofia and Macron Prague, Ltd. While the Bulgarian system remained in more or less a demonstrator stage only, the Czech one satisfied Macron's requirements as to syntactic coverage. However, Macron expressed serious worries about the speed of the system, should this be really introduced to the market. Following this, several possibilities of using finite-state automata (FSA) as means for speeding up the performance of the system were designed, developed and implemented, in particular:

- for detecting sentences where none of the predefined errors can occur (thus ruling out such sentences from the procedure of error-search proper)
- for detecting which one(s) of the predefined error types might possibly occur in a particular sentence (hence, cutting down the search space of the error-search proper)
- for detecting errors which are of such a nature that their occurrence might be discovered by a machinery simpler than full-fledged parsing with constraint relaxation
- for splitting (certain cases of) complex sentences into independent clauses, allowing thus for the error-detection to be performed on shorter strings.

2 Lexicalization of Error Search

Very many of the errors to be discovered by the system can be traced down to mismatches of (values of) features projected into the syntactic structure from the lexicon. Even though the error searching capabilities of the system are not limited in principle to these lexically induced errors, for a practical implementation it turned out to be useful to narrow down the error search of the system to almost only these kinds of errors, for the following reasons:

1. the loss of generality of the system is in fact only minimal, since the majority of errors which the system is able to detect are of this nature anyway (the only exception being agreement errors involving NPs with complicated internal structure, e.g., ellipses or coordination)
2. this loss of error coverage (almost negligible for a real application) is outweighed by substantial gain in overall (statistical) speed of the system, which is achieved by adding a preprocessing phase consisting of a finite state automaton passing through the input string and looking for a lexical trigger of a contingent error:

[1] As for the alternative technology, cf. (Holan, Kuboň, and Plátek, 1997)

- if this automaton does not find any such trigger, the time-consuming grammar-checking process proper (i.e. parsing, possibly also reparsing with relaxed constraints) is not started at all and the sentence is immediately marked as one containing no detectable error
- if this automaton finds such a lexical trigger of an error, it 'remembers' its nature so that in the following phases, only the respective constraints are relaxed (which helps to cut down the search space, as compared to reparsing with relaxing of all predefined-errors-related constraints)

As an example of this idea, let us consider a system dealing with errors in subject-verb agreement in Czech (and taking - for the very purpose of this example - detection of no other errors into account). Since the realistic part of such errors in Czech is the '-I/-Y' dichotomy on homophonic past tense verb endings occurring on plural verbs ('-I' ending standing with plural masculine animate subjects, '-Y' ending with plural masculine inanimate and feminine subjects), the preprocessing finite-state automaton marks all sentences not containing any of these forms (i.e. all sentences containing only singular verbs, or plural verbs but in present tense or in neuter gender, or infinite verb forms) as 'containing no detectable error', without any actual grammar-checking taking place (it is, however, obvious that this does not necessarily mean that the sentences are truly correct - they just do not contain the kind of error the system is able to detect).

3 Alternative Error-Classification and Error Search by Finite Automata

Another important step towards the application of FSA to error-detection was developing a new dimension of classification of errors to be detected: apart from the more standard criteria of frequency and performance/competence, we developed a scale based on the complexity of the formal apparatus needed for the detection of the particular error type (as for error typology developed for the purpose of the error detection techniques used in the project, cf. (Rodriguez Selles, Galvez, and Oliva, 1996)). On the one end of this scale were errors recognizable within a strictly local context, such as commas missing in front of a certain kind of complementizers (subordinating conjunctions) or incorrect vocalization of a preposition (in both Bulgarian and Czech, certain prepositions ending normally with a consonant get a supporting vocal in case the word that follows them also starts with a consonant - the parallel in English would be the opposition between the two forms *a* and *an* of the indefinite article). On the other end of the scale we put, e.g., the general case of subject-verb agreement errors. Practically more important was the question whether there exists a class of errors with complexity of detection lying between the "trivial errors" and the errors for the detection of which a full-fledged analysis is necessary - in other words, the question whether there exist some errors for the recognition of which

- on the one hand, a limited local context is insufficient (i.e. it is necessary for this end to process a substring of length which cannot be set in advance, in general the whole input string),
- on the other hand, it is not necessary to use the power of the full-fledged parser, and, in particular, it is sufficient to use the power of a finite state automaton or only slight augmentation thereof.

Following some linguistic research, two such error types have been selected for implementation, and while one of them is just a marginal subtype of an error in subject-verb agreement, the other is an error type of its own, and in addition one of really crucial importance for practical grammar-checking due to its high frequency of occurrence.

The former error to be detected by the finite state machinery is a particular instance of an error where a plural masculine animate subject is conjoined with a verb in a plural feminine form (cf. also the example above). The idea of detecting some particular cases of this error by a finite state automaton results from the combination of the following observations:

- the nominative plural form of masculine animate nouns of the declension types *pán* and *předseda* is not ambiguous (homonymous) with any other case forms (apart form vocative case, which we shall deal with immediately below); this means that if such a form occurs in a sentence, then this form can be only
  - either a subject,
  - or a nominal predicate (with copula)
  - or a comparison to these, adjoined by means of the conjunctions *jako, jakožto* or *coby*
  - or an exclamative expression (in nominative or vocative case)
- due to rules of Czech interpunction, any exclamative expression has to be separated from the rest of the sentence by commas
- also, due to rules of Czech interpunction, two finite verbs in Czech must be separated from each other by either a comma or by one of the following coordinating conjunctions: *a, i* and *nebo*

Hence, if we build up a finite state automaton able to recognize the following substrings:

1. <unambiguous masculine animate noun in nominative plural> followed by any string containing neither a finite verb form nor a comma

nor one of the conjunctions *a, i* and *nebo* followed by <unambiguous past participle in plural feminine>

or (due to free word order)

2. <unambiguous past participle in plural feminine> followed by any string containing neither a finite verb form nor a comma nor one of the conjunctions *a, i, nebo* followed by <unambiguous masculine animate noun in nominative plural> and combine it with a simple automaton able to detect the absence of the words *jako, jakožto* and *coby* as well as the absence of any finite form of the copula *být* ('to be') in the sentence, then we may conclude that we have built a device able to detect whether a sentence contains a particular instance of a subject-verb agreement violation.

The detection of the latter error is also based on the Czech interpunction rule prescribing that there always must occur a comma or a coordinating conjunction between two finite verb forms. Hence, a simple finite state automaton checking whether between any two finite verb forms a comma or a coordinating conjunction occurs is able to detect many cases of the omission of a comma at the end of an embedded subordinated clause, which is one of the most frequent errors at all. (Of course, the word-forms of the verb must be unambiguously identifiable as such - i.e. such forms as *žena, jedu, tratím, holí* etc., do not qualify due to their part of speech ambiguity, which means that in sentences containing them this strategy cannot be used).

4 Using FSA for Splitting a Sentence into Clauses

The last idea how to gain efficiency is that of splitting the sentence (if possible) into clauses before the processing, which has a two-fold positive effect on the overall process of grammar-checking:

1. it is less time consuming to parse two 'shorter' strings than one longer (assuming that parsing is at least cubic in time, this follows trivially from the inequality

   $$A^3 + B^3 < A^3 + 3A^2B + 3AB^2 + B^3 = (A+B)^3$$

   for A,B positive - length of strings)

2. it is possible to detect an error in one of the substrings (clauses) irrespective to the results of analysis of (any of) the other one(s); in particular, also in cases where at least one of them was not analyzed and, hence, also the parsing (including the parsing with relaxed constraints) of the whole input could not have been performed on the original string, which would have hindered the error message pertinent to the substring successfully parsed during the parsing with constraint relaxation to be issued.

In particular, this means that measures are to be found which would allow for splitting the input sentence into clauses by purely superficial criteria. Obviously, this is not possible in all cases (for all sentences), but on the other hand it is also clear that in any language there exists a (statistically) huge subset of sentences of this language where such techniques are applicable. For Czech, such an approach might be implemented using pattern matching techniques which would recognize for example the following patterns (and use them in an obvious way for splitting the sentence into clauses):

1. <any string> <finite verb> <any string> <conjunctive coordinating conjunction> <any string> <finite verb> <any string> <end of sentence>
2. <any string> <finite verb> <any string> <comma> <non-conjunctive coordinating conjunction or complementizer> <any string> <finite verb> <any string> <end of sentence>
3. <complementizer> <any string> <finite verb> <any string> <comma> <any string> <finite verb> <any string> <end of sentence>

where the expressions have the following meaning(s):

- <any string> is a variable for any string not containing elements of the following nature: finite verb or word form homonymous with a finite verb, coordinating conjunction (of any kind), complementizer, any interpunction sign
- <finite verb> is a variable
  - for a main verb (not for an auxiliary) specified for person,
  - or for a past participle of a main verb;

  neither of these might be homonymous in part of speech (but they might be ambiguous within the defined class - such verbs as *podrobí, proudí* do qualify)
- <end of sentence> is simply either a full-stop, a question-mark, an exclamation-mark, a colon or a semi-colon.

All the remaining expressions have clear mnemonics, and also the classes which they stand for do not contain elements which are ambiguous as to part of speech.

5 Significance and Caveats

The techniques to be used for gaining overall performance, speed and memory efficiency etc., of a grammar-checking system presented result solely from research concerning relevant properties of the syntax of a particular language (Czech, in part also Bulgarian), and, hence, they are strongly language-dependent. However, it seems to be self-evident that the core idea is transferable to other languages. The

introduction of these techniques contributes to the process of ripening of the system into a real industrial application at least in the following points:

- it speeds up the overall performance of the system considerably (in the order ranging from one to two magnitudes, depending on the text to be processed) by avoiding full-fledged parsing to be performed in unnecessary cases or by making this parsing simpler
- it extends its coverage, in particular the capabilities of the system to recognize as error-free also a large number of sentences which in the original version of the system would be unanalyzable by the non-relaxed grammar (as well as by the grammar with relaxed constraints, for that matter) due to either incompleteness of the grammar proper or the exhaustion of hardware resources.

There is a serious caveat to be issued, however: since they do not employ full analysis of the input sentence, these techniques are - albeit probably only rarely on the practical level - more likely to issue incorrect error messages, in the sense that their capabilities of detecting an erroneous sentence are *exactly* the same as on the full-fledged approach, but their capabilities of detecting *what* kind of error occurred in the sentence are slightly reduced. For example, in (the Czech equivalent of) the sentence

*Your wife drives very drives fast

a grammar-checker based solely on the full-fledged philosophy would correctly recognize that the same verb is repeated twice, while a checker using only finite state automaton detecting the presence/absence of a comma or a coordinating conjunction between two finite verbs issues a message concerning exactly the 'missing comma' - and similar examples can be constructed also for all the other cases. In other words, there is a price to be paid for the speed-up of the error-checking process by means of the techniques proposed.

References

Holan, T., V. Kuboň, and M. Plátek. 1997. A prototype of a grammar checker for Czech. In this volume.

Rodriguez Selles, Y., M.R. Galvez, and K. Oliva. 1996. Error detection techniques in grammar-checking. Technical report of the PECO2824 project, Autonomous University of Barcelona and University of Saarland.