# Detecting Grammar Errors in Children’s Writing: A Finite State Approach

Sylvana Sofkova Hashemi
Department of Linguistics, Göteborg University
sylvana@ling.gu.se

**ABSTRACT**

This paper reports on the development of a finite state system for finding grammar errors in Swedish text written by children. The writing problems are more frequent for this group and the distribution of the error types is different from texts written by adults. The detection approach involves subtraction of automata that represent two “positive” grammars with varying degree of detail. The difference between the automata corresponds to the search for writing problems that violate the grammars. The constituents of various fragments are identified by the use of lexical-prefix-first strategy and application of incremental parsing. While the grammatical coverage is still rather small, yielding rather large numbers of false alarms, the technique can be applied to agreement phenomena, verb selection phenomena and some word order phenomena. The aim is to include also detection of missing sentence boundaries.

## Introduction

Research and development of grammar checking techniques has been carried out since the 1980’s, mainly for English and also for other languages, e. g. French (Chanod, 1993), Dutch (Vosse, 1994), Czech (Kirschner, 1994), Spanish and Greek (Bustamente and León, 1996). In the case of Swedish, the development of grammar checkers started not until the later half of the 1990’s with several independent projects, resulting in the first product release in November 1998 - *Grammatifix* (Arppe, 2000; Birn, 2000), now part of the Swedish Microsoft Office 2000.

The work described here is a continuation of a project from this starting period, that explores the use of finite state techniques for grammar checking. The approach differs from the other Swedish projects not only in the choice of technique used, but also in that the grammatical errors are found without any description of the erroneous patterns. The detection process involves partial parsing as writing of ”positive” grammars at different accuracy levels that as transducers can be subtracted from each other. The difference between the automata corresponds to the search for writing problems that violate the grammars. Karttunen et al (1996) use this technique to find instances of invalid dates.

The current system, using the *Xerox Finite-State Tool* (XFST) (Karttunen et al, 1997), is divided in four main modules: the lexicon lookup, the grammar, the parser and the error finder. A simple emacs environment is used both for testing and development of finite state grammars. The environment shows the results of an XFST process run on the current emacs buffer in a separate buffer. An XFST mode allows for menus to be used and recompile files in the system.

## The Corpus Data

The analyses of writing problems is based on a corpus of 31 756 words (3 361 word types), composed of (mostly) computer written and hand written essays written by children between 9 to 13 year old. In general, the text structure of the compositions reveals clearly the influence of spoken language and performance difficulties in spelling, segmentation of words, the use of capitals and punctuation, varying both by individuals and age.

In total, 306 grammatical error instances were found in the 134 narratives. The most recurrent grammar problem concerns the omission of finite verb inflection (28%), i. e. when the main finite verb in a clause is in infinitive form and the appropriate present or past tense endings are dropped (a quite usual phenomena in spoken Swedish). Other more frequent grammar problems are wrong choice of a pronoun (15%), wrong pronoun case (8%), extra or missing words (11%), errors in verb chains (6%) and agreement in noun phrases (5 %).

Punctuation problems are also included in the analyses. In general, the use of punctuation varies from no usage at all (mostly among the youngest children) to rather sparse marking. The omission of end of sentence marking is quite obvious problem (35%), but the most frequent punctuation problem concerns the comma (81%).

## The Lexicon Lookup

The lexicon is built as a finite state transducer, using the Xerox tool *Finite-State Lexicon Compiler* (Karttunen, 1993). It takes a string and maps inflected surface form to a tag containing part-of-speech and feature information, e. g. applying the transducer to the string *kvinna* ‘woman’ will return *[nn utr sin ind nom]*. The word analyser is based on *LEXIN* (58 326 word forms; Skolverket, 1992) and the lexicon developed in Daniel Ridings’ lexicon project in the Language Technology Programme, HSFR/NUTEK (100 000 word forms). The network does not handle unknown words. The size of the lexicon is rather small, but the LEXIN-part includes valuable information on valency (see further in Andersson et al, 1998 and 1999).

The system uses a simple lookup without any disambiguation. The reason for that is that the disambiguation heuristics of a tagger may fail with a text that contains errors, because the information needed for the detection of errors is (often) filtered out (see Cooper and Sofkova Hashemi, 1998). The strategy of a lexical lookup which leaves all lexical information present without attempting any disambiguation seems to be the most safe strategy in order to ensure that no information needed is lost.

## The Grammar and Parsing

The grammar module is further subdivided into a *broad grammar* and a *narrow grammar,* that include regular expressions reflecting truths about the grammatical structure of Swedish, differing in the level of detail. For example the simple regular expression in the *broad* grammar:

```
define VC [Verb Adv* Verb (Verb)];
```

recognises potential verb clusters (both grammatical and ungrammatical), consisting of a sequence of two or three verbs in combination with some adverbs (zero or more). On the other hand, the following rules in the *narrow* grammar take into account the internal structure of a verb cluster, i. e. the rules define the grammar of (modal or temporal) auxiliary verbs followed by optional sentence adverb(s) and main verb:

| | | |
|---|---|---|
| define VC1 | [Mod Adv* VerbInf]; | *kan (inte) springa*<br>"can (not) run " |
| define VC2 | [Mod Adv* PerfInf VerbSup]; | *skall (inte) ha sprungit*<br>"will (not) have run[sup]" |
| define VC3 | [Perf  Adv* VerbSup]; | *har (nog) sprungit*<br>"have (not) run[sup]" |
| define VC4 | [Perf  Adv* ModSup VerbInf]; | *har (inte) velat springa*<br>"have (not) want[sup] run[sup]" |

The various kinds of constituents are marked out in a text using the lexical-prefix-first method, i. e. parsing first from left margin of a phrase to the head and then extending the phrase by adding on complements.

The actual parsing (based on the broad grammar definitions) is incremental similarly to the methods described in Ait-Mokhtar and Chanod (1997). With this method, the parsing results vary depending on the order in which the various grammatical facts are applied. The system recognises the higher phrases in the first phase (e. g. vp, pp, np) and then applies the second phase in the reverse order (e. g. np, pp, vp). This method gives a greater efficiency and flexibility to the system by decreasing the size of the nets when compiling and that (some) false parses can be blocked.

## Error Detection

The error finder is a separate module in the system which means that the grammar and parser could be used directly in a different application. The nets of this module correspond to the difference between the two grammars, broad and narrow. For example, the regular expression:

[ ”<vc>” [VC - VCgram] ”</vc>” ]

where VC is the part of the broad grammar describing broad structure of a verb cluster and VCgram is the part of the narrow grammar describing the grammar of auxiliary verbs, will find verb clusters that violate these constraints within the VC-boundaries that have been previously marked out by applying the broad grammar.

So far the technique was used to detect agreement errors in noun phrases, verb selection phenomena (in particular selection of finite and non-finite verb forms in main and subordinate clauses and infinitival complements) and local word order phenomena (e. g. placement of negation). Also some attempts were maid to detect missing sentence boundaries, starting with clause and verb subcategorisation and trying to make use of the valency information stored in the lexicon of the system.

Further, it is possible to use this method to perform error diagnostics. This is achieved by subtraction of small parts of the narrow grammar representing specific constraints that can be violated.

## Evaluation

The grammatical coverage of the system is still rather small, yielding a large number of false alarms. There are in general two kinds of false alarms that occur: either because of the “smallness” of grammar for the particular constituents occurring in the phrase, or due a false parse depending either on the ambiguity of constituents or the “wideness” of the parse, i. e. too many constituents are included when applying the longest-match strategy. The following example shows an ambiguous parse:

*Linda , brukade ofta vara i stallet.*[1]
”Linda , used to be often in the stable.”

**<Error finite verb> <vp><vpHead><np>Linda </np></vpHead></vp> </Error>** , <vp><vpHead> <vc> brukade ofta <np>vara </np> </vc> </vpHead><pp><ppHead>i </ppHead><np>stallet </np></pp></vp>.

[1] The example is authentic, including the comma after the proper name ’Linda’, that also influence the parsing result, i. e. without the comma ’Linda’ would be parsed as part of a verb cluster than a separate verb phrase.

where ‘linda’ is a proper name and parsed as a noun phrase, but it also is an (infinite) verb in infinitive (*to wrap*) and will then be marked as a verb phrase. Then the error finder marks this phrase as a finite verb error, due the violation of this constraint in the narrow grammar that does not allow any infinite verbs without the presence of a preceding (finite) auxiliary verb or infinitival marker ‘att’. Next example shows the effect of the longest-match strategy:

*I hålet som pojken hade hittat fanns en mullvad.*
“In the hole that the boy had found was a mole.”

<pp><ppHead>I </ppHead><np>hålet </np></pp> som <np>pojken </np><vp><vpHead> **<Error verb after Vaux> <vc>hade hittat fanns </vc></Error>** </vpHead><np>en mullvad </np></vp>.

where the verb cluster boundary is too wide, including both a verb cluster and a finite verb belonging to the next clause (here missing the appropriate punctuation marking).

Increasing the coverage of the positive grammar will solve some problems with false marking, but also some kind of filtering techniques (e. g. finite state intersection grammar; Tapanainen, 1997) should be involved in the solution of false parsing. However, involving filtering of any kind may also cause that the errors in the text may not be found.

## Conclusion

The simple finite state technique of subtraction presented in this paper, has the advantage that the grammars one needs to write to find errors are always positive grammars rather than grammars written to find specific errors. This means that they will find a large class of errors without having to specify them individually. Although the grammatical coverage of the system is rather small, the technique can be applied to detect some kinds of grammar errors. Most of the false alarms depend on the smallness of the grammar, other occur due the ambiguity of the constituents resulting in false parses. Further expansion of the grammatical coverage and some other filtering techniques are needed in order to block the false alarms.

In conclusion, the robustness and modularity of this system makes it possible to perform both error detection and diagnostics and that the grammars can be reused for other applications that do not necessarily have anything to do with error detection, e. g. for educational purposes, speech recognition, and for other users such as dyslectics, aphasics, deaf and foreign speakers.

## References

Ait-Mokhtar, S. and Chanod, J-P. (1997) Incremental Finite-State Parsing. In *Proceedings of ANLP'97*, Washington March 31st to April 3rd, pp. 72-79

Andersson, R., Cooper, R. and Sofkova Hashemi, S. (1998) Finite State Grammar for Finding Grammatical Errors in Swedish Text: a finite-state word analyser, Report-9808. Department of Linguistics, Göteborg University.

Andersson, R., Cooper, R. and Sofkova Hashemi, S. (1999) Finite State Grammar for Finding Grammatical Errors in Swedish Text: a system for finding ungrammatical noun phrases in Swedish text, Report-9903. Department of Linguistics, Göteborg University.

Arppe, A. (2000) Developing a grammar checker for Swedish. In *Proceedings of the 12th Nordic Conference in Computational Linguistics, Nodalida´99*. Department of Linguistics, Norwegian University of Science and Technology, Trondheim, pp. 13-27.

Birn, J. (2000) Detecting grammar errors with Lingsoft's Swedish grammar checker. In *Proceedings of the 12th Nordic Conference in Computational Linguistics, Nodalida´99*. Department of Linguistics, Norwegian University of Science and Technology, Trondheim, pp. 28-40.

Cooper, R. and Sofkova Hashemi, S. (1998) Finite State Grammar for Finding Grammatical Errors in Swedish Text: Report-9809. Department of Linguistics, Göteborg University.

Chanod, J-P. (1993) A Broad-Coverage French Grammar Checker: Some Underlying Principles. In *Proceedings of the Sixth International Conference on Symbolic and Logical Computing*. Dakota State University Madison, South Dakota.

Bustamente, F. R. and León, F. S. (1996) GramCheck: A Grammar and Style Checker. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, pp. 175-181.

Karttunen, L. (1993) Finite-State Lexicon Compiler. Technical Report ISTL-NLTT-1993-04-02, Xerox Palo Alto Research center, Palo Alto, California.

Karttunen, L., Chanod, J-P., Grefenstette, G. and Schiller, A. (1996) Regular Expressions for Language Engineering. In Natural Language Engineering 2 (4), pp. 305-328.

Karttunen, L., Gaál, T. and Kempe, A. (1997) Xerox Finite-State Tool. Technical Report Version 6.0.4, Xerox Research Centre Europe, Grenoble, Meylan, France.

Kirschner, Z. (1994) Czecker – a Maquette Grammar-Checker for Czech. In *The Prague Bulletin of Mathematical Linguistics 62*, Prague: Universita Karlova.

Skolverket (1992) Lexin: språklexikon för invandrare. Stockholm: Nordstedts Förlag.

Tapanainen, P. (1997) Applying a Finite-State Intersection Grammar. In Roche, E. and Schabes, Y. (eds.) *Finite-State Language Processing*, The MIT Press.

Vosse, T. (1994) *The Word Connection. Grammar-Based Spelling Error Correction in Dutch.* Doctoral Dissertation. Enschede: Neslia Paniculata.