Ambiguity Resolution by Reordering Rules in Text Containing Errors

Sylvana Sofkova Hashemi
Department of Linguistics, Göteborg University
Box 200, SE-405 30 Göteborg, SWEDEN
sylvana@ling.gu.se

Abstract

Writing aids such as spelling and grammar checkers are often based on texts by adult writers and are not sufficiently targeted to support children in their writing process. This paper reports on the development of a writing tool based on a corpus of Swedish text written by children and on the parsing methods developed to handle text containing errors. The system uses finite state techniques for finding grammar errors without actually specifying the error. The 'broadness' of the grammar and the lexical ambiguity in words, necessary for parsing text containing errors, also yields ambiguous and/or alternative phrase annotations. We block some of the (erroneous) alternative parses by the order in which phrase segments are selected, which causes bleeding of some rules and more 'correct' parsing results are achieved. The technique shows good coverage results for agreement and verb selection phenomena.

1 Introduction

Writing on a computer in school often involves making a fair copy from a handwritten draft. Although a computer is an excellent means for the writing process, especially the linguistic tools are not used adequately. Spelling and grammar correctors are in general developed for and adapted to adult writers and have difficulties to support children in their writing development and give no space for acquisition or training. Errors in texts written by school children are more frequent and the distribution of the error types is different from adult writers.

This paper reports on the development of a finite state system for finding grammar errors, called *FiniteCheck*, based on a corpus of Swedish text written by school children. The system applies descriptions of correct language use in the detection process of grammatical violations and contains no rules describing the nature of the erroneous segments the system searches for. The approach (following Karttunen et al., 1996) for finding errors involves developing automata that represent two 'positive' grammars with varying degree of detail and then subtracting the detailed one from the general one. The difference between the automata corresponds to a grammar for errors.

2 Grammar Checkers

2.1 Current Systems

Whereas *spelling checkers* are standard in most word processors, grammar checking is a rather recent technology, especially for Swedish. Different methods and techniques have been applied to handle *nonsense words* and thus operate on isolated words as most spelling correctors do. Both statistical and rule-based methods and also algorithms that to some extent take into consideration the surrounding context (i.e. context-sensitive errors) or how a word is pronounced have been used for spelling correction (cf. Kukich, 1992).

Grammar checkers involve techniques and solve problems above the single word level and require syntactic, semantic or even discourse analysis (see Section 2.2). Grammar checking techniques started to develop first in the 1980's with products mainly for English (see Jensen et al, 1993; Vernon, 2000) but also for other languages, e.g. French (Chanod, 1996), Dutch (Vosse, 1994), Czech (Kirschner, 1994), Spanish and Greek (Bustamente and León, 1996). Computer-based grammar checking for Swedish is fairly recent and has primarily focused on the needs of adult writers. The first product release of such a writing aid was in November 1998 with the tool *Grammatifix* (Arppe, 2000; Birn, 2000), now part of the Swedish Microsoft Office 2000. Two other research groups developed grammar checking prototypes: *Granska* (Knutsson, 2001; Domeij, 2003) and *Scarrie* (Sågvall Hein, 1999).

2.2 Methods and Techniques

Many of the grammar checking systems are commercial products and technical documentation is often minimal or even absent. *Critique* (known until 1984 as *Epistle*) is an exception, a system developed in collaboration with IBM within the *Programming Language for Natural Language Processing* (PLNLP) project (Jensen et al., 1993). This tool is based on a parser using *Augmented Phrase Structure Grammar* (ACFG) and produces a complete analysis for all sentences (even ungrammatical) by application of *relaxation* rules when parsing fails on the first try or *parse fitting* procedure identifying the head and its constituents (Heidorn, 1993; Jensen et al., 1993). This approach of providing analysis of all sentences had influenced other grammar formalisms such as *Constraint Grammar* (Karlsson et al., 1995) or *Functional Dependency Grammar* (Järvinen and Tapanainen, 1998). The methods of *rule relaxation* and *parse fitting* had an impact on the development of other grammar checking systems.

The three Swedish tools use different technology to analyze unrestricted text and detect grammar errors. The lexical analysis in *Grammatifix* is based on the morphological analyzer *SWETWOL*, designed according to the principles of *two-level morphology* (Karlsson, 1992). The part-of-speech assignment applies the *Swedish Constraint Grammar* (SWECG), a surface-syntactic parser applying context-sensitive disambiguation rules (Birn, 1998). Errors are detected by *partial parsing* and *relaxation* on rules, regarding certain word sequences as phrases despite grammar errors in them.

Granska combines *probabilistic* and *rule-based* methods, where specific error rules (around 600) and local applied rules detect ungrammaticalities in free text. The lexical analyzer applies *Hidden Markov Models* and a rule matching system analyses the tagged text searching for grammatical violations defined in the detection rules and produces error description and a correction suggestion for the error (Carlberger & Kann, 1999).

The grammar checker in *Scarrie* is based on a previously developed parser, the *Uppsala Chart Parser* (UCP), a procedural, bottom-up parser, applying a longest path strategy (Sågvall Hein, 1983). The parsing strategy of erroneous input is based on *constraint relaxation* and application of *local error rules*. The grammar is in other words underspecified to a certain level, allowing feature violations and parsing of ungrammatical word sequences (Wedbjer Rambell, 1999).

The Swedish approaches to detection of grammar errors vary from chart-based methods in *Scarrie*, application of constraint grammars in *Grammatifix*, to a combination of probabilistic and rule-based methods in *Granska*. *Scarrie* and *Granska* identify erroneous patterns by partial analysis, whereas *Grammatifix* produces full analysis for both grammatical and ungrammatical sentences. All the tools define (wholly or to some extent) explicit error rules describing the nature of the error they search for. In the process of error detection they either proceed sentence by sentence, requiring recognition of sentence boundaries, or they rely in their rules on for instance capitalization conventions.

2.3 Error Coverage

Current grammar checking systems are restricted to a small set of all possible writing errors, concerning mostly syntactic analysis. The choice of what types of errors are detected in the Swedish tools is based on analysis of errors in writing of certain groups of writers (e.g. professional writers, writers at work). The coverage of error types is very similar between the systems, including errors in noun phrase agreement and agreement in predicative complement, pronoun case after preposition, word order, errors in verbs, etc.

Observations with children writing on a computer in school (Hård af Segerstad & Sofkova Hashemi, 2006; Sofkova Hashemi, forthcoming) and performance tests of the Swedish tools on texts written by school children (Sofkova Hashemi, 2003) show that grammar checkers do not sufficiently support school children in their writing development. The grammatical mistakes found in texts written by children display different frequency and distribution than in adults and the text structure as whole is different. Main clauses are often joined together without conjunctions and punctuation marks often delimit larger textual units than syntactic sentences. Sentence boundaries and capitalization are something the Swedish tools rely on in their detection process, which may have impact on the coverage results. Although the systems cover many of the types of errors found in school texts, they detect around 12% of all writing errors (Sofkova Hashemi, 2003) (see Section 6). Performance on text data such as newspaper texts and student compositions evaluated within the frames of the separate projects shows a much higher coverage of error detection on average 58% (Birn, 2000; Knutsson, 2001; Sågvall Hein et al., 1999).

3 The Training Data

3.1 The Child Data Corpus

FiniteCheck, the grammar error detector reported in this paper, is based on a corpus of Swedish text written by school children. This *Child Data* corpus of 29 812 words (3 373 word types) is composed of computer written and hand written essays written by children between 9 and 13 years of age. 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, with fairly wide variation both by individual and age. In total, 260 instances of grammatical errors were found in 134 narratives.

3.2 The Error Types

The most frequent grammatical violation concerns *the omission of finite verb inflection* (42% of all errors), i.e. when the main finite verb in a clause lacks the appropriate present or past tense endings:

(1) *På natten* ****vakna*** *jag av att brandlarmet tjöt*
in the-night wake[untensed] I from that fire-alarm howled
– In the night I woke up from that the fire-alarm went off.

The correct form of the verb *vakna* 'wake' should be in the past tense, i.e. *vaknade* 'woke'. This type of error arises from the fact that the writing is highly influenced by spoken language. In spoken Swedish regular weak verbs in the past tense often lack the appropriate ending and the spoken form then coincides with the infinitive (and for some verbs also imperative) form of the verb.

Other frequent grammar problems concern *extra inserted or missing words* in sentences (22%), here the preposition *i* 'in' is missing:

(2) *Gunnar var på semester* ****_ norge*** *och åkte skidor*.
Gunnar was on vacation _ Norway and went skis
– Gunnar was on vacation in Norway and skied.

word choice errors (11%), here the verb *att vara lika* 'to be alike' requires the particle *till* 'to' in combination with the noun phrase *sättet* 'the-manner' and not *på* 'on' as the writer uses:

(3) *vi var väldigt* ***lika *på sättet***
we were very like on the-manner
– We were very alike in the manners.

errors in *noun phrase agreement* (6%), here the correct form of the noun phrase requires the noun to be definite as in *den närmsta handduken* 'the nearest towel':

(4) *jag tar* ***den närmsta *handduk*** *och slänger den i vasken*
I take the[def] nearest[def] towel [indef] and throw it in the sink
– I take the nearest towel and throw it into the sink.

errors in *verb chains* (3%), here the auxiliary verb should be followed by an infinitive, *ska bli* 'will become', but in this case the present tense is used:

(5) *Men kom ihåg att det inte* ***ska *blir*** *någon riktig brand.*
but remember that it not will becomes[pres] some real fire
– But remember that there will not be a real fire.

Other grammar errors occurred less than ten times in the whole corpus, including reference errors, agreement between subject and predicative complement, definiteness in single nouns, pronoun form, errors in infinitive phrases, word order. 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. In the following example the main clauses are joined together and the boundary between the sentences is not marked:

(6) *nasse blev arg han gick och la sig med dom andra syskonen.*
nasse became angry he went and lay himself with the other siblings
– Nasse got angry. He went and lay down with the other siblings.

The finite verb problem, verb form in verb chains and infinitive phrases and agreement problems in noun phrase are the four types of errors detected by the current system, *FiniteCheck*.

4 System architecture

The framework for detection of grammar errors in *FiniteCheck* is built as a network of finite state transducers compiled from regular expressions including operators defined in the *Xerox Finite State Tool* (XFST) (Karttunen et al., 1997). Each automaton in the network composes with the result of previous application and in principle all the automata can be composed into a single transducer.

There are in general two types of transducers in use: one that annotates text in order to select certain segments and one that redefines or refines earlier decisions. Annotations of any kind are handled by transducers defined as *finite state markers* that add reserved symbols into text and mark out syntactical segments, grammar errors, or other patterns aimed at selections. *Finite state filters* are used for refinement and/or revision of earlier decisions.

The system runs under UNIX in a simple *Emacs* environment used 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 sequenced finite state transducers of *FiniteCheck* are divided in four main modules: *the lexicon lookup, the grammar, the parser* and *the error finder* – see Figure 1.

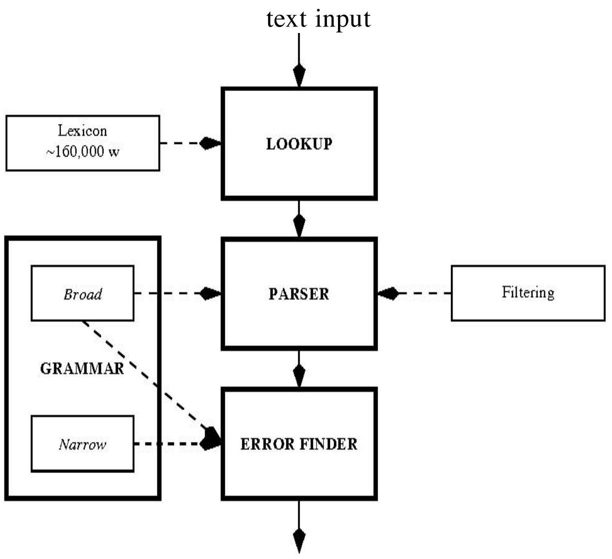

Figure 1: The system architecture

4.1 The Lexicon Lookup

The lexicon of around 160, 000 word forms, is built as a finite state transducer, using the Xerox tool *Finite State Lexicon Compiler* (Karttunen, 1993). The lexicon is composed from two resources and 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 morphosyntactic tags follow directly the relevant string or token. More than one tag can be attached to a string, since no contextual information is taken into account. The morphosyntactic information in the tags is further used in the grammars of the system. The set of tags follows the *Stockholm-Umeå Corpus* project conventions (Ejerhed et al, 1992), including 23 category classes and 29 feature classes that were extended with 3 additional categories. Below is an example of a lookup on the example sentence in (5):

(7) Men[kn] kom[qmvb prt akt][vb prt akt] ihåg[ab][pl] att[sn][ie] det[pn neu sin def sub/obj] [dt neu sin def] inte[ab] ska[vb prs akt][mvb prs akt] blir[vb prs akt] någon[dt utr sin ind][pn utr sin ind sub/obj] riktig[jj pos utr sin ind nom] brand[nn utr sin ind nom]

4.2 The Grammar

The grammar module includes two grammar sets with (positive) rules reflecting the grammatical structure of Swedish, differing in the level of detail. The *broad grammar* is especially designed to handle text with ungrammaticalities and the linguistic descriptions are less accurate accepting both valid and invalid patterns. The *narrow gramar* is fine and accurate and accepts only the grammatical segments. For example, the regular expression in (8) belongs to the broad grammar set and recognizes potential *verb clusters* (VC) (both grammatical and ungrammatical) as a pattern consisting of a sequence of two or three verbs in combination with (zero or more) adverbs:

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

This automaton accepts all the verb cluster examples in (9), including the ungrammatical instance (9c) (marked by an asterisk '*'), where a finite verb follows a (finite) auxiliary verb.

(9) a. *kan inte springa* 'can not run'
   b. *skulle ha sprungit* 'would have run [sup]'
   c. **ska blir* 'will be [pres]'

Corresponding rules in the narrow grammar set represented by the regular expressions in (10) take into account the internal structure of a verb cluster and define the grammar of modal auxiliary verbs (Mod) followed by (zero or more) adverb(s), and either a verb in infinitive form (VerbInf) as in (10a), or a temporal verb in infinitive (PerfInf) and a verb in supine form (VerbSup), as in (10b). These rules thus accept only the grammatical segments in (9) and will not include example (9c). The actual grammar of grammatical verb clusters is a little bit more complex.

```
(10) a. define VC1 [Mod Adv* VerbInf];
     b. define VC2 [Mod Adv* PerfInf VerbSup];
```

4.3 The parser

The various kinds of constituents are marked out in a text using a *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 in a similar fashion as the methods described in Ait-Mohtar and Chanod (1997), where the output from one layer serves as input to the next, building on the segments. The system recognizes the head phrases in certain order in the first phase (verbal head, prepositional head, adjective phrase) and then applies the second phase in the reverse order and extends the phrases with complements (noun phrase, prepositional phrase, verb phrase). The parsing method is described in detail in Section 5.

4.4 Error Detection

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

By subtracting the narrow grammar from the broad grammar we create machines that will find ungrammatical phrases in a text. For example, the regular expression in (11) identifies verb clusters that violate the narrow grammar of modal verb clusters (VC1 or VC2, defined in (10)) by subtracting these rules from the more general (overgenerating) rule in the broad grammar (VC, defined in (8)) within the boundaries of a verb cluster ('<vc>', '</vc>'), that have been previously marked out in the parsing stage.

```
(11) define VCerror [ "<vc>" [VC - [VC1 |
                              VC2]] "</vc>" ];
```

By application of a marking transducer in (12), the found error segment is annotated directly in the text as in example (13).

```
(12) define markVCerror [VCerror ->
          "<Error verb after Vaux>" ... "</Error>"];
```

(13) Men <vp> <vpHead> kom ihåg </vpHead> </vp> att <np> det </np> <vp> <vpHead> inte **<Error verb after Vaux>** <vc> ska blir </vc> **</Error>** </vpHead> <np> någon <ap> riktig </ap> brand </np> </vp>

5 Parsing

5.1 Parsing procedure

The rules of the (underspecified) *broad grammar* are used to mark syntactic patterns in a text. A *partial, lexical-prefix-first, longest-match, incremental* strategy is used for parsing. The parsing procedure is *partial* in the sense that only portions of text are recognized and no full parse is provided for. Patterns not recognized by the rules of the (*broad*) grammar remain unchanged. The maximal instances of a particular phrase are selected by application of the *left-to-right-longest-match replacement* operator.

The segments are built on in cascades in the sense that first the heads are recognized, starting from the left-most edge to the head (so called *lexical-prefix*) and then the segments are expanded in the next level by addition of complement constituents. The regular expressions in (14) compose the marking transducers of separate segments into a three step process.

```
(14)    define parse1[markVPhead .o.
                 markPPhead  .o. AP];
        define parse2 [markNP];
        define parse3 [markPP .o. markVP];
```

First the verbal heads, prepositional heads and adjective phrases are recognized by composition in that order (*parse1*). This output serves then as input to the next level, where the adjective phrases are extended and noun phrases are recognized and marked (*parse2*). This output in turn serves as input to the last level, where the whole prepositional phrases and verb phrases are recognized in that order (*parse3*). During and after this parsing annotation, some phrase types are further expanded with post-modifiers, split segments are joined and empty results are removed.

The 'broadness' of the grammar and the lexical ambiguity in words, necessary for parsing text containing errors, also yields ambiguous and/or alternative phrase annotations. We block some of the (erroneous) alternative parses by the order in which phrase segments are selected, which causes bleeding of some rules (i.e. the parsing order destroys application of another parsing rules; a feature mostly used of the ordering of phonological rules) and more 'correct' parsing results are achieved. The order in which the labels are inserted into the string influences the segmentation of patterns into phrases. Further ambiguity resolution is provided for by filtering automata.

5.2 The Heuristics of Parsing Order

Reordering rules used in parsing allows us to resolve certain ambiguities. For example, marking verbal heads before noun phrases will prefer a verb phrase interpretation of a string over a noun phrase interpretation and avoid merging constituents of verbal heads into noun phrases and yielding noun phrases with too-wide range.

For instance, marking first the sentence in (15) for noun phrases will interpret the pronoun *De* 'they' as a determiner and the verb *såg* 'saw', that is exactly as in English homonymous with the noun 'saw', as a noun and merges these two constituents to a noun phrase as shown in (16). *De såg* will subsequently be marked as ungrammatical, since a number feature mismatch occurs between the plural *De* 'they' and singular *såg* 'saw'.

(15) *De såg ledsna ut*
they looked sad out
- They seemed sad.

(16) <np>De såg </np> <np>ledsna </np> ut .

Composing the marking transducers by first marking the verbal head and then the noun phrase will instead yield the more correct parse. Although the alternative of the verb being parsed as verbal head or a noun remains (i.e. *såg* 'saw' is still tagged as a noun in a noun phrase), the pronoun *De* 'they' is now marked correctly as a separate noun phrase and not merged together with the main verb into a noun phrase:

(17) <np> De </np> <vpHead> <np> såg </np> </vpHead> <np> ledsna </np> ut .

The output at this stage is then further refined and/or revised by application of *filtering* transducers. Earlier parsing decisions depending on lexical ambiguity are resolved (e.g. adjectives parsed as verbs) and phrases extended (e.g. with postnominal modifiers). Other structural ambiguities, such as verb coordinations or clausal modifiers on nouns, are also taken care of.

This ordering strategy is not absolute however, since the opposite scenario is possible where parsing noun phrases before verbal heads is more suitable, as for instance in example (18) below, where the string *det öppna fönstret* 'the open window' will be split in three separate noun phrase segments when applying the order of parsing verbal heads before noun phrases, due the homonymity between an adjective and an infinitive or imperative verb form (19).

(18) *han tittade genom det öppna fönstret*
he looked through the open window
- He looked through the open window

(19) <np> han </np><vpHead> tittade </vpHead> genom <np> det </np> <vpHead> <np> öppna </np> </vpHead> <np> fönstret </np>

We analyzed the ambiguity frequency in the *Child Data* corpus and found that occurrences of nouns recognized as verbs are more frequent than the opposite. On this ground, we chose the strategy of marking verbal heads before marking noun phrases. In the case of the opposite scenario, the false parsing can be revised and corrected by an additional filter (see Section 5.3).

A similar problem occurs with homonymous prepositions and nouns. For instance, the string *vid* is ambiguous between an adjective ('wide') and a preposition ('by') as shown in example (20) and influences the order of marking prepositional heads and noun phrases. Parsing prepositional heads before noun phrases is more suitable for preposition occurrences as shown in (22) in order to prevent the preposition from being merged as part of a noun phrase, as in (21):

(20) *Jag satte mig vid bordet*
I sat me by the-table
– I sat down at the table.

(21) <np> Jag </np> satte <np> mig </np> <np> <ppHead> vid </ppHead> bordet </np>

(22) <np> Jag </np> satte <np> mig </np> <ppHead> <np> vid </np> </ppHead> <np> bordet </np>

5.3 Further Ambiguity Resolution

Nouns, adjectives and pronouns are homonymous with verbs and might then be interpreted by the parser as verbal heads. Adjectives homonymous with prepositions can be analyzed as prepositional heads. These parsing decisions can be redefined at a later stage by application of *filtering* transducers.

As exemplified in (19) above, the consequence of parsing verbal heads before noun phrases may yield noun phrases that are split into parts, due to the fact that adjectives are interpreted as verbs. The filtering transducer in (23) adjusts such segments and removes the erroneous (inner) syntactic tags (i.e. replaces them with the empty string '0') so that only the outer noun phrase markers remain and converts the split phrase in to one noun phrase yielding (24).

```
(23) define adjustNPAdj [
        "</np><vpHead><np>" -> 0 || Det _ APPhr
        "</np></vpHead>" NPPhr,,
        "</np></vpHead><np>"   ->   0   ||   Det
        "</np><vpHead><np>" APPhr _ ];
```

(24) <np> han </np> <vpHead> tittade </vpHead> genom <np> det öppna fönstret </np>

The regular expression consists of two replacement rules that apply in parallel. They are constrained by the surrounding context of a preceding determiner (Det) and a subsequent adjective phrase (APPhr) and a noun phrase (NPPhr) in the first rule, and a preceding determiner and an adjective phrase in the second rule.

5.4 Parsing Expansion and Adjustment

The text is now annotated with syntactic tags and some of the segments have to be further expanded with *postnominal attributes* and *coordinations*. In the current system, partitive prepositional phrases are the only postnominal attributes taken care of. The reason is that grammatical errors were found in these constructions.

By application of the filtering transducer in (25) the example text in (26) with the partitive noun phrase *en av dom gamla husen* 'one of the old houses' split into a noun phrase followed by a prepositional head that includes the partitive preposition *av* 'of' and yet another noun phrase from the parsing stage (27) is merged to form a single noun phrase, as shown in (28). This automaton removes the redundant inner syntactic markers by application of two replacement rules, constrained by the right or left context. The replacement occurs simultaneously by application of parallel replacement.

(25) define adjustNPPart [
"</np><ppHead>" -> 0 || _ PPart
"</ppHead><np>",,
"</ppHead><np>" -> 0 ||
"</np><ppHead>" PPart _];

(26) *Virginia hade öppnat en tygaffär i en av dom gamla husen*.
Virginia had opened a fabric-store in one of the old houses[def].
- Virginia had opened a fabric-store in one of the old houses.

(27) <np> Virginia </np> <vp><vpHead> <vc> hade öppnat </vc> </vpHead> <np> en tyg-affär </np> i <np> en </np> <ppHead> av </ppHead> <np> dom <ap> gamla </ap> husen </np> .

(28) <np> Virginia </np> <vp> <vpHead> <vc> hade öppnat </vc> </vpHead> <np> en tyg-affär </np> i <NPPart> en av dom <ap> gamla </ap> husen </np>

Other filtering transducers are used for refining the parsing result and eliminate incomplete parsing decisions such as prepositional heads without a following noun phrase.

6 The System Performance

6.1 Result on Child Data

The implemented error detector, *FiniteCheck*, cannot at present be considered as a fully developed grammar checking tool, but still even with its restricted lexicon and small grammar the results are promising. So far the technique was used to detect agreement errors in noun phrases, selection of finite and non-finite verb forms in main and subordinate clauses and infinitival complements. The implementation proceeded in two steps. In the first phase we devoted all effort to detection of the grammar errors, working mostly with the errors and not paying much attention to the text as a whole. The second phase involved blocking of the resultant false alarms found in the first stage.

In *Table 1* we show the final results of error detection in the training corpus of Child Data. There were altogether 15 agreement errors in noun phrase, 110 errors in the form of finite verb, 7 errors in the verb form after an auxiliary verb and 4 errors in verbs after infinitive marker.

Error type	No. Errors	CA	FA	R	P	F
Agreement in NP	15	15	62	100%	19%	33%
Finite verb form	110	96	126	87%	43%	58%
Verb form after aux. verb	7	6	47	86%	11%	20%
Verb form after inf. marker	4	4	0	100%	100%	100%
Total	136	121	235	89%	34%	49%

Table 1. Performance of *FiniteCheck* on Child Data: correct alarms (CA), false alarms (FA), recall (R), precision (P), F-value (F).

FiniteCheck detected all the agreement errors in noun phrases and all erroneous verb forms after an infinitive marker, only a portion of other errors in verb form was missed. The precision of the system is rather low, primarily due the ambiguity of the texts and the number of alarms marking other errors such as segmentation or spelling errors. This side-effect is difficult to eliminate totally and gives rather rise to new questions of how to handle also these types of writing problems that concern spelling rather than grammar.

The three Swedish grammar checkers mentioned above in Section 2: *Grammatifix*, *Granska* and *Scarrie*, have been tested on the *Child Data*. The result of their performance is shown in *Figure 2*, below, together with the results of *FiniteCheck*.

These three tools are designed to detect errors in text different from the nature of the *Child Data* and thus not surprisingly the accuracy rates are in overall low. The total recall rate for the four error types covered by *FiniteCheck* is between 9% and 21% in these three tools and precision varies between 16% to 35%. Errors in noun phrases seem to be better covered than verb errors.

In the case of *Grammatifix*, errors in verbs are not covered at all. Half of the noun phrase errors were identified and only five errors in the finite verb form. *Granska* covered all four error types and detected at most half of the errors for three of these types. However, only seven instances of errors in finite verb form were identified. *Scarrie* had difficulties with errors in verb form after infinitive marker that were not detected at all. Errors in noun phrase were the best detected type.

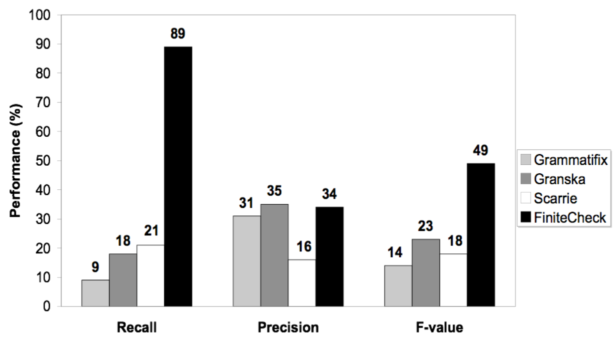

Figure 2: Performance of All Systems on Child Data

The detection performance of these three tools on Child Data is in general half that good in comparison to our detector and the fact that the error type with worst coverage (finite verbs) is the one most frequent among children indicates clearly the need for specialized grammar checking tools for children.

6.2 Result on Text Written by Adult

The current system was also tested on a text of 1 070 words written by an adult, one of the demonstration texts used by *Granska*. The performance of *FiniteCheck* on this text is presented in *Table 2*. We found 17 noun phrase agreement errors, 5 errors in the form of finite verb and 1 error in the verbform after an auxiliary verb in the text. *FiniteCheck* found all the verb form errors and most of the agreement errors, ending in a recall value of 87%. False alarms occurred also only in the agreement errors, resulting in a precision rate of 71% and an F-value of 78%.

Error type	No. Errors	CA	FA	R	P	F
Agreement in NP	17	14	6	82%	70%	76%
Finite verb form	5	5	1	100%	83%	91%
Verb form after aux. verb	1	1	1	100%	50%	67%
Total	23	20	8	87%	71%	78%

Table 2. Performance of *FiniteCheck* on Text Written by Adult: correct alarms (CA), false alarms (FA), recall (R), precision (P), F-value (F).

The three Swedish grammar checkers were also tested on this adult text, that reflects more the text type these tools are designed for. The results presented in *Figure 3* show an average recall rate of 52% for the three Swedish grammar checkers, *FiniteCheck* scored 87%. These tools had difficulties to detect the verb form errors, whereas most of the errors in noun phrase agreement were found. The opposite scenario applies for precision, where *FiniteCheck* had slightly worse rate (71%) than *Grammatifix* and *Granska*, which had a precision above 90%. *Scarrie's* precision was 65%. In the combined measure of recall and precision (F-value) our system obtained 78%, which is slightly better in comparison to the other tools that had 70% or less in F-value.

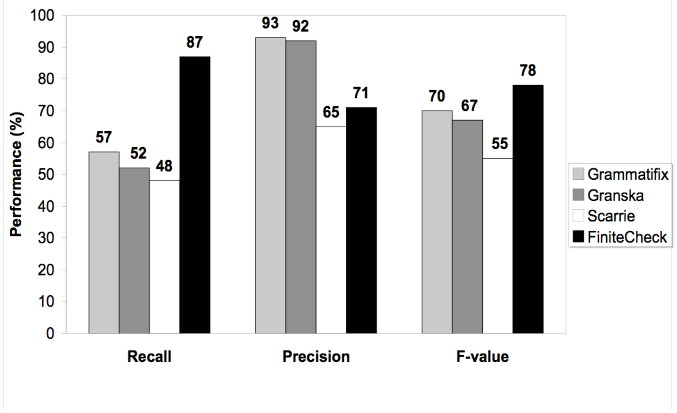

Figure 3: Performance of All Systems on Text Written by Adult

7 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. Thus, covering the valid rules of language means that the rule sets remain quite small and practically no prediction of errors is necessary.

The approach aimed further at minimal information loss in order to be able to handle text containing errors. The degree of ambiguity is maximal at the lexical level, where we choose to attach all lexical tags to strings. At higher levels, structural ambiguity is treated by parsing order, grammar extension and some other heuristics. There is an essential problem of ambiguity resolution on complement decisions that remains to be solved. Sequences of words grammatical in one context and ungrammatical in another are treated the same. The system overinterprets and gives rise to false alarms, mostly due the application of longest-match, but more seriously information indicating an error may be filtered out by erroneous segmentation and errors overlooked. A 'higher' mechanism is needed in order to solve these problems that takes into consideration the complement distribution and solves these structural dependencies.

The linguistic accuracy of the system is comparable to other Swedish grammar checking tools, that actually performed worse on the Child Data. The low performance of the Swedish tools on Child Data motivates clearly the need for adaptation of grammar checking techniques to children. The other tools obtained in general much lower recall values and although the error type of particular error was defined, the systems had difficulties to identify the errors, probably due problems to handle such a disrupted structure with many adjoined sentences and high error frequency.

Further, 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-Mohtar, Salah and Chanod, Jean-Pierre (1997) Incremental Finite-State Parsing, In *ANLP'97,* Washington, pp. 72-79.

Arppe, A. (2000) Developing a Grammar Checker for Swedish.In *The 12th Nordic Conference of Computational Linguistics,* NODALIDA-99, pp.13-27.

Birn, J. (1998). *Swedish Constraint Grammar: A Short Presentation.* [http://www.lingsoft.fi/doc/swecg/].

Birn, J. (2000) Detecting Grammar Errors with Lingsoft's Swedish Grammar Checker. In *The 12th Nordic Conference of Computational Linguistics,* NODALIDA-99, pp.28-40.

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

Carlberger, J. and Kann, V. (1999) Implementing an efficient part-f-speech tagger. *Software – Practice and Experience,* 29(9):815-832.

Chanod, J.-P. (1996) A Broad-Coverage French Grammar Checker: Some Underlying Principles. In *The Sixth International Conference on Symbolic and Logical Computing,* Dakota State University Madison, South Dakota.

Domeij, R. (2003). *Datorstödd språkgranskning under skrivprocessen. Svensk språkkontroll ur användarperspektiv.* Doktorsavhandling, Stockholms Universitet, Institutionen för lingvistik.

Ejerhed, E., Källgren, G., Wennstedt, O. and Åström, M. (1992) *The Linguistic Annotation System of the Stockholm-Umeå Corpus Project.* Report 33. University of Umeå, Department of Linguistics.

Heidorn, G. (1993). Experience with an easily computed metric for ranking alternative parses. In Jensen, K., Heidorn, G., and Richardson, S. D. (eds.) *Natural Language Processing: The PLNLP Approach*. Kluwer Academic Publishers, Dordrecht.

Hård af Segerstad, Y. and Sofkova Hashemi, S. (2006) Learning to Write in the Information Age: A Case Study of Schoolchildren's Writing in Sweden. In Van Waes, L., Leijten, M. och Neuwirth, C. (eds.), *Writing and Digital Media*, Elsevier.

Jensen, K., Heidorn, G. and Richardsson, S. D. (eds.) (1993) *Natural Language Processing: The PLNLP Approach.* Kluwer Academic Publishers, Dordtrecht.

Järvinen, T. and Tapanainen, P. (1998). Towards an implementable dependency grammar. In Kahane, S. and Polguere, A. (eds.) The *Proceedings of COLIN-*

GACL'98, Workshop on 'Processing of Dependency-Based Grammars', pages 1–10. Universite de Montreal, Canada.

Karlsson, F. (1992). SWETWOL: Comprehensive morphological analyzer for Swedish. *Nordic Journal of Linguistics*, 15:1–45.

Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A. (1995). *Constraint Grammar: a language-independent system for parsing unrestricted text.* Mouton de Gruyter, Berlin.

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., Grefenstette, G. and Schiller, A. (1996) Regular Expressions for Language Engineering, In *Natural Language Engineering 2 (4)* 305-328.

Karttunen, L., Gaál, T. and Kempe, A. (1997) *Xerox Finite State Tool*. Xerox Research Centre Europe,

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

Knutsson, O. (2001). *Automatisk språkgranskning av svensk text.* Licentiatavhandling, KTH, Institutionen för numerisk analys och datalogi, Stockholm.

Kukich, K. (1992) Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, Vol. 24, No. 4: 377 - 439.

Sofkova Hashemi, S. (2003) *Automatic Detection of Grammar Errors in Primary School Children's Texts. A Finite State Approach.* Doctoral dissertation. Gothenburg Monographs in Linguistics 24. Department of Linguistics, Göteborg University.

Sofkova Hashemi, S. (forthcoming) *The role of writing aid in the text production of school children*. Department of Linguistics, Göteborg University

Sågvall Hein, A. (1983). *A Parser for Swedish. Status Report for SveUcp*. (UCDLR-83-2). Uppsala University, Department of Linguistics. February 1983.

Sågvall Hein, A. (1999) *A grammar checking module for Swedish.* Report from the Scarrie-project: DEL 6.6.3, June 1999, Dept. of Linguistics, Uppsala University.

Sågvall Hein, A., Olsson, L.-G., Dahlqvist, B., and Mats, E. (1999). Evaluation report for the Swedish prototype. In Sågvall Hein, A. (ed.) *Reports from the SCARRIE project,* Deliverable 8.1.3, June 1999. Uppsala University, Department of Linguistics.

Vernon, A. (2000) Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities. *Computers and Composition* 17, 329-349.

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

Wedbjer Rambell, O. (1999). Swedish phrase constituent rules. A formalism for the expression of local error rules for Swedish. In Sågvall Hein, A. (ed.) *Reports from the SCARRIE project.* Uppsala University, Department of Linguistics.