The Development and Performance of a Grammar Checker for Swedish: A Language Engineering Perspective

J. Carlberger, R. Domeij, V. Kann, O. Knutsson
KTH Nada, SE-100 44 Stockholm, Sweden
E-mail: {jfc,domeij,viggo,knutsson}@nada.kth.se.

(*Received 16 December 2004*)

Abstract

This article describes the construction and performance of Granska – a surface-oriented system for grammar checking of Swedish text. With the use of carefully constructed error detection rules, written in a new structured rule language, the system can detect and suggest corrections for a number of grammatical errors in Swedish texts. In this article, we specifically focus on how erroneously split compounds and disagreement are handled in the rules.

The system combines probabilistic and rule-based methods to achieve high efficiency and robustness. The error detection rules are optimized using statistics of part-of-speech bigrams and words in a way that each rule needs to be checked as seldom as possible.

We have found that the Granska system with higher efficiency can achieve the same or better results than systems with conventional technology.
Keywords: grammar checking, part-of-speech tagging, error detection rules, optimization, hidden Markov models.

1 Introduction

Grammar checking is one of the most widely used tools within language technology. Spelling, grammar and style checking for English have been an integrated part of common word processors for about twenty years now. Some well-documented systems are Epistle/Critique and the grammar checker in Word97 for English (Jensen *et al.* 1983; Jensen, Heidorn, and Richardson 1993; Heidorn 2000), and the rule-based system for Dutch by Vosse (1994) and ReGra for Brazilian Portuguese (Martins *et al.* 1998). Most grammar checkers are based on handcrafted rules, however a few statistical and machine learning approaches to general error detection have been tried during the years (see for example Atwell (1987) and Izumi (2003)).

Current research seems to focus mostly on machine learning techniques for so-called context sensitive spelling checking, and in particular on confusions sets. (see for example Mangu (1997), Golding (1999), and Carlson (2001)).

For languages with a smaller number of speakers, such as Swedish, advanced

tools have been lacking. Recently, the first grammar checker for Swedish, developed by the Finnish company Lingsoft, was launched in Word 2000 (Arppe 2000). This grammar checker is partly based on the Swedish constraint grammar SWECG. There are also two research prototypes available for Swedish, Scarrie (Sågvall Hein 1998) which includes a advanced parser and a system using a finite state approach called FiniteCheck (Sofkova Hashemi, Cooper, and Andersson 2003).

In this article, another grammar checker for Swedish is presented. This grammar checker, called GRANSKA has been designed and developed with efficiency and robustness in focus. One important goal has been to make GRANSKA robust against texts with many errors without limiting the feedback given. Other goals have been to develop GRANSKA in a cost-effective way, and to test and use it in user studies.

GRANSKA is a hybrid system that uses surface grammar rules to check grammatical constructions in Swedish. The system combines probabilistic and rule-based methods to achieve high efficiency and robustness, which is a necessary prerequisite for a grammar checker that runs in real time in direct interaction with users (Kukich 1992). Using error detection rules, the system can detect a number of Swedish grammar problems and suggest corrections for them.

2 About the Errors

When developing a phenomena-based grammar checker an error catalog has to be defined. It can for instance focus on theoretically interesting error types, or error types which are frequent in one specific user group. It can also be based on studies of language use in general. The error catalog of GRANSKA is partly based on studies on frequent errors (see Wedbjer (1999) and Sofkova Hashemi (2003) for overviews of grammatical errors in Swedish), but also on the descriptions of correct grammatical constructions, and on decisions of which errors that are plausible to detect with the technology chosen. Although the error catalog of GRANSKA includes many error types, we focus on three types of errors in the following.

1. Split compounds is an error that is increasing in Swedish. It is also interesting because of its limited description in the literature.
2. Internal NP disagreement is quite common in Swedish texts, and grammatical Swedish NPs is also a well-studied field, and their ungrammatical counterparts.
3. Disagreement errors between the subject and the predicative involves long distance dependencies, which should test the limits of current tools for text analysis included in Granska.

2.1 Erroneously Split Compounds

Swedish is a compounding language where lexemes can be written together as a single word with two or more components e.g. *mansröst* with the lexemes *man* (man) and *röst* (voice). These constructions are called closed compounds. This is a common way of constructing the head of an NP, but it is also possible to

construct an NP with the same lexemes with a genitive construction and with two separated words e.g. *mans röst* (eng. man's voice). Syntactically, they are forming a more or less equivalent NP. Semantically they differ, and that is the main problem with erroneously split compounds – to know if the split compound can construct a compound that is semantically plausible, and therefore should not be signalled as an error.

If a split compound creates an ungrammatical structure it must be an error; but if the construction is grammatical but for humans not acceptable, it should ideally also be signalled as an error.

As well as in English (with mostly open compounds), Swedish compounds cannot be listed – compounds are the most frequent hapax words – and a lexical approach is thus out of question. The semantics of a compound normally constructs one obvious interpretation, and one more rare. The last part of the compound gives the word its word class, but not necessary its semantic interpretation. The relations between the lexemes in the compound are complex.

2.2 Agreement Errors in Swedish

Frequent agreement errors in Swedish are located within the noun phrase. The words in an NP can disagree according to gender, number and definiteness. Other agreement errors appear between the subject and the predicative, which can involve long distance dependencies. Even more problematic agreement errors concern anaphoric agreements between phrases within the sentence or between sentences. Granska as well as the grammar checker of MS Word try to find agreement errors in NPs and disagreement between the subject and the predicative. However, both grammar checkers focus mainly on agreement errors within NPs.

Agreement errors within the NP are more problematic to detect than one can expect. The Swedish language is full of acceptable constructions that permit disagreement. The following phenomena can illustrate how different constructions can cause false alarms if they are not carefully treated. NPs with predicative attributes like *statsrådet ensamt* (eng. the cabinet minister alone) can either be constructed where with both words' gender values in neuter or as a construction where the gender value of *statsrådet* (cabinet minister) is based on the fact that *statsrådet* is a person, and persons normally have the gender value common in Swedish. Hence, the adjective *ensam* in common gender in the construction *statsrådet ensam* is also allowed in Swedish. Adjectives in superlative must also be carefully handled as in *Jag kan utan den största ansträngning motstå frestelsen* (eng. I can resist the temptation without the greatest effort). The problem here is that the determiner *den* is in definite form in spite of the indefinite form of the head noun. Another complication is definiteness in the detection of disagreement in NPs with restrictive relative clauses as in *Den vän som jag en gång hade fanns inte mer* (eng. The friend I once had is gone.), where the noun *vän* (friend) in indefinite form is grammatical despite the definite determiner *den* (the).

If the rules are only using local information to detect errors, different kind of attributes in NPs can cause false alarms, for instance *Han tillhörde ett gatans par-*

lament (eng. He belonged to the parliament of the street) The word *gatans* (the street's) is an attribute to the head noun *parlament* (parliament), and the phrase has an obligatory agreement between the determiner *ett* (a) and *parlament* (parliament), and not between the adjacent words *ett* and *gatans* which is the normal case. There are many more constructions in NPs that must be treated in a grammar checker for Swedish, and the main problem is avoiding them without reducing the recall of the error type more than necessary.

3 The Granska System

We will first present the structure of Granska, and then in more detail describe four important parts of the system: the part-of-speech (PoS) tagging module, the construction of error detection rules, the algorithms for rule matching, and the generation of error corrections. Finally, we describe the performance of tagging and error detection.

In Figure 1, the modular structure of Granska is presented. First, in the tokenizer, potential words and special characters are recognized as such. In the next step, a tagger is used to assign disambiguated part of speech and inflectional form information to each word (Carlberger and Kann 1999). The tagged text is then analyzed by surface grammar rules that find structures such as noun phrases. This information is then sent to the error rule matching component where error rules are matched with the text in order to search for specified grammatical problems. The error rule component also generates error corrections and instructional information about detected problems that are presented to the user in a graphical interface. In addition, the system contains a spelling error detection and correction module, called Stava that can handle Swedish compounds (Domeij, Hollman, and Kann 1994; Kann *et al.* 2001). Incorporating Stava in Granska improves both spelling and grammar checking. For example, unknown proper names are not signalled as spelling errors if the tagger in Granska identifies them as such, and spelling corrections that do not fit in the context are not proposed. Stava can also be used from inside the error rules, e.g. in the process of checking split compound errors, see below.

The Granska system is implemented in C++ under Unix, and it has recently been integrated in a language learning environment called Grim[1].

4 Part-of-Speech Tagging

In PoS tagging of a text, each word and punctuation mark in the text is assigned a morphosyntactic tag. We have designed and implemented a tagger based on a second order Hidden Markov Model (Charniak *et al.* 1993; Charniak 1996). Given a sequence of words $w_{1..n}$, the model finds the most probable sequence of tags $t_{1..n}$ according to the equation

[1] Grim is a freely available web client that works under most operating systems. See the web page of Grim: `http://www.nada.kth.se/grim/`

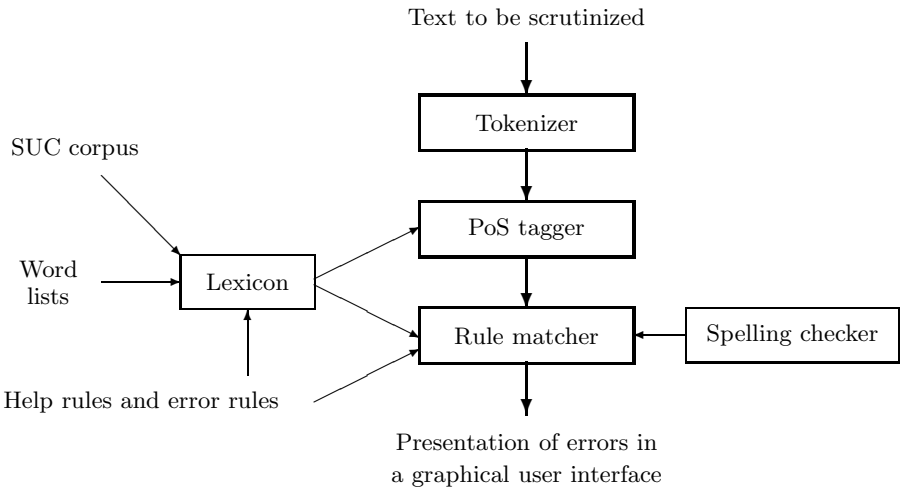

Fig. 1. An overview of the GRANSKA system.

$$\mathcal{T}(w_{1..n}) = \arg\max_{t_{1..n}} \prod_{i=1}^{n} P(t_i|t_{i-2}, t_{i-1}) P(w_i|t_i). \quad (1)$$

Estimations of the two probabilities in this equation are based on the interpolation of relative counts of sequences of 1, 2 and 3 tags and word-tag pairs extracted from a large tagged corpus.

For unknown words, we use a statistical morphological analysis adequate for Swedish and other moderately inflecting languages. This analysis is based on relative counts of observed tags for word types ending with the same 1 to 5 letters. This captures both inflections (tense *-ade* in *hämtade* (fetched)) and derivations (nounification *-ning* in *hämtning* (pick-up)). Similarly, if the first letter of the word is upper case the probability of proper noun is increased.

We also perform an analysis that finds the last word form of compounds, which are common in Swedish. The possible tags of the last word form indicate possible tags (and probability estimation) for an unknown compound word. These two analyses are heuristically combined to get estimations of $P(w_i|t_i)$, which enables unknown words to work in the model. This method combines morphological information for unknown words with contextual information of surrounding words, and resulted in a tagger that tags 97% of known and 93% of unknown words correctly using a tag set of size 140. For more information, see (Carlberger and Kann 1999). We have found that nearly all tags in the tag set are necessary in order to detect the errors searched for.

The objective of GRANSKA is to find grammatical errors in a text, but how can a ungrammatical text be tagged? For example, should the adjective *glada* (happy), which has the same form in singular and plural in *en glada dagar* (a happy days) be tagged as singular or plural, or as both? Is the disagreement with the noun *dagar*

(days) more important than with the determiner *en* (a)?. We have found that it is almost always better to choose one of the taggings, since if *glada* is tagged as singular then the error rules will detect *glada dagar* as an agreement error, and if *glada* is tagged as plural then *en glada* will also be detected as an agreement error. Thus it is better to disambiguate even when it is not clear how to do it.

5 Error Rules

The error rule component uses carefully constructed error rules to process the tagged text in search for grammatical errors. Since the Markov model also disambiguates and tags morphosyntactically deviant words with only one tag, there is normally no need for further disambiguation in the error rules in order to detect an error. An example of an agreement error is **en** *litet hus* (a small house), where the determiner *en* (a) does not agree with the adjective *liten* (small) and the noun *hus* (house) in gender. The strategy differs from most rule-based systems which often use a complete grammar in combination with relaxation techniques to detect morphosyntactical deviations (see for example Vosse (1994) and Sågvall (1998)).

The error rules of Granska are expressed in a new and general rule language developed for this project (Knutsson 2001). It is partly object-oriented and has a syntax resembling Java or C++. An error rule in Granska that can detect the agreement error in *en liten hus*, is shown in Rule 1 below.

Rule 1

```
cong22@incongruence {
   X(wordcl=dt),
   Y(wordcl=jj)*,
   Z(wordcl=nn & (gender!=X.gender | num!=X.num | spec!=X.spec))
-->
   mark(X Y Z)
   corr(X.form(gender:=Z.gender, num:=Z.num, spec:=Z.spec))
   info("The determiner" X.text "does not agree with the noun" Z.text)
   action(scrutinizing)
}
```

Rule 1 has two parts separated with an arrow. The first part contains a matching condition. The second part specifies the action that is triggered when the matching condition is fulfilled. In the example, the action is triggered when a determiner is found followed by a noun (optionally preceded by one or more (`*`) attributes) that differs (`!=`) in gender, number or (`|`) species from the determiner. Each line in the first part contains an expression that must evaluate to true in a matching rule. This expression may be a general expression (with all standard boolean and numeric operators) and may refer to values (matching texts, word classes, or features) of the earlier parts of the rule.

The action part of the rule first (in the `mark` statement) specifies that the erroneous phrase should be marked in the text. Then (in the `corr` statement) a function is used to generate a new inflection of the article from the lexicon, one that agrees with the noun. This correction suggestion (in the example **ett** *litet hus*) is presented

to the user together with a diagnostic comment (in the `info` statement) describing the error.

In most cases, the tagger succeeds in choosing the correct tag for the deviant word on probabilistic grounds (in the example *en* is correctly analyzed as an indefinite, singular and common gender article by the tagger). However, since errors are statistically rare compared to grammatical constructions, the tagger can sometimes choose the wrong tag for a morphosyntactically deviant form. In some cases when the tagger is known to make mistakes, the error rules can be used in re-tagging the sentence to correct the tagging mistake. An example of this is when the distance between two agreeing words is larger than the scope of the tagger. Thus, a combination of probabilistic and rule-based methods is used even during basic word tag disambiguation.

We use *help rules* (functions, possibly recursive) to define phrase types that can be used as context conditions in the error rules. In rule 2 below, two help rules are used in detecting agreement errors in predicative position. The help rules specify that copula should be preceded by an NP followed by one or more (+) PPs.

Rule 2

```
pred2@predicative {
   T(wordcl!=pp),
   (NP)(),
   (PP)()+,
   X(wordcl=vb & vbt=kop),
   Y(wordcl=jj & (gender!=NP.gender | num!=NP.num)),
   Z(wordcl!=jj & wordcl!=nn)
-->
   mark(all)
   corr(if NP.spec=def then
        Y.form(gender:=NP.gender, num:=NP.num, spec:=ind) else
        Y.form(gender:=NP.gender, num:=NP.num) end)
   info("The noun phrase" NP.text "does not agree with the adjective" Y.text)
   action(scrutinizing)
}

NP@ {
   X(wordcl=dt)?,
   Y(wordcl=jj)*,
   Z(wordcl=nn)
-->
   action(help, gender:=Z.gender, num:=Z.num, spec:=Z.spec, case:=Z.case)
}

PP@ {
   X(wordcl=pp),
   (NP)()
-->
   action(help, gender:=NP.gender, num:=NP.num, spec:=NP.spec, case:=NP.case)
}
```

The help rules in the example are specified in the subroutines `NP@` and `PP@` which

define noun phrases and prepositional phrases respectively. These subroutines are called from the higher level rule for predicative agreement (`pred2@predicative`). Note that the help rule `PP@` uses the other help rule `NP@` to define the prepositional phrase. In the action part the help rules return the features of the phrases.

The help rules make the analysis approach the analysis of a phrase structure grammar. Help rules make it possible for the system to perform a local phrase analysis selectively, without parsing other parts of the sentence that are not needed in the detection of the targeted error type. Thus, by calibrating the level of analysis that is needed for the case at hand, the system obtains high efficiency. The possibility to call help rules makes the rule language more powerful than languages based on regular expressions, such as XFST (Beesley and Karttunen 2003) or CG (Karlsson 1990).

Above, we have shown how agreement errors are handled in the system. Now we will turn to the error type of erroneously split compounds. So far, we have mainly focussed on erroneously split compounds of the type noun+noun which stand for about 70% of the various types (Domeij, Knutsson, and Öhrman 1999).

Detection of erroneously split compounds where the first part cannot stand alone, forming a non-existing word, is trivial with a good spelling checker. The error detection is done by listing those first parts in the lexicon and classifying them so that an error rule can be made to search the text for such a first part in combination with any other noun, as for example in *pojk byxor* where *pojk* is the first part form of *pojke* (boy) which is combined with *byxor* (trousers).

In other cases, when both parts have the form of full words, the strategy for detecting erroneously split compounds makes use of the fact that the first noun, unlike the last, must be uninflected (indefinite and singular). Since the combination of an uninflected noun followed by any noun is a slightly unusual syntactical combination in grammatically correct sentences, it can be used to find candidates for split compound errors. Other contextual cues like disagreement and agreement are also used. Disagreement between the preceding determiner and the first part of the split, and agreement between the determiner and the last part are useful hints to locate split compounds safely. Before signaling the error, the candidates are checked against a spelling checker for compound recognition. If the spelling checker recognizes the compound as such, the two nouns in the text are marked as a split compound and the corrected compound is given as a suggestion alternative.

We have also found that rules for clause boundary recognition could increase recall and precision even more, especially for split compounds. Therefore, we have experimented with rules based on Ejerhed's clause boundary recognition algorithm (Ejerhed 1999). By applying the error rules for split compounds only on, for example, clauses without ditransitive verbs (with only one NP after the verb), Granska can avoid false alarms and still detect errors in another clause within the same sentence.

Many errors can be difficult to detect because of ambiguities that are irresolvable on contextual grounds only. One example is *en exekverings enhet* (an execution unit). The first noun *exekvering* belongs to a group of nouns that take an *-s* when compounded with another noun (*exekvering-s+enhet*). When the compound is er-

roneously split, the form of the first noun coincides with the genitive form (an execution's unit), which has the same syntactical distribution as the error construction and therefore cannot be distinguished from the split compound case.

6 Rule Matching

Presently, there are about 180 error detection rules, 60 help rules and 110 accepting rules (rules that sort out exceptions) in GRANSKA. This could be compared to the Swedish grammar checker of MS Word containing about 650 error rules (Birn 2000). Each rule in GRANSKA may be matched at any position (i.e. word) in the text, and there may even exist several matchings of a rule with the same starting position and of different length. The rule matcher tries to match rules from left to right, evaluating the expression of each token in the left hand side of the rule, and stopping as soon it finds out that the rule cannot be matched.

The rule language allows the operators `*` (zero or more), `+` (one or more) and `?` (zero or one) for tokens, and `;` (or) between rules. Together with the possibility of writing possibly recursive help rules, this makes the rule language a context free language. The results of all calls to help rules are cached (memorized) so if the same help rule is called in several rules at the same position it will only need to be computed once. Standard parsing techniques like LL(1) or LALR(1) (Aho, Sethi, and Ullman 1984) cannot be used due to the complexity of the constraints that can be formulated in the rule language and due to the fact that we want to find all matchings, not just a single parse tree.

It is inefficient to try to match each error rule at each position in the text. We therefore perform a statistical optimization, where each rule is analyzed in advance. For each position in the rule, the possible matching words and taggings are computed. In fact, the possible tag bigrams for each pair of positions are computed. Then, using statistics on word and tag bigram relative frequencies, the position of the rule that is least probable to match a Swedish text is determined. The least probable position is stored in the field `leastProbablePos` of the rule object.

This means that this rule is checked by the matcher *only* at the positions in the text where the words or tag bigrams of this least probable position in the rule occur. For example, a noun phrase disagreement rule may require a plural adjective followed by a singular noun in order to match. Such tag combinations are rare, and with this optimization approach, only the small portion of word sequences in a text containing this tag combination will be inspected by this rule.

It is important to note that this optimization does not miss any matchings and is fully automatic. GRANSKA preprocesses the rule set by detecting the optimal positions in each rule and stores two tables representing this information on disk. The first table, `bigramRulesToCheck`, describes, for each tag bigram, which rules that should be checked when that tag bigram occurs in the text. The second table, `wordRulesToCheck`, contains the words appearing in the rules and describes, for each word, which rules that should be checked when that word occurs in the text. The rule matching algorithm is as follows:

```
FindOptimizedMatchings(AbstractSentence sen)
    rulesToCheck← ∅
    for i ← 1 to sen.length−1 do
        foreach Rule r ∈ bigramRulesToCheck[sen.tags[i],sen.tags[i + 1]] do
            startPos ← i−r.leastProbablePos
            if startPos> 0 and startPos+r.minScope≤sen.length then
                rulesToCheck.Add(r,startPos)
    for i ← 1 to sen.length do
        foreach Rule r ∈ wordRulesToCheck[sen.word[i]] do
            startPos ← i−r.leastProbablePos
            if startPos> 0 and startPos+r.minScope≤sen.length then
                rulesToCheck.Add(r,startPos)
    for (Rule r, int startPos) ∈ rulesToCheck do
        r.TryMatching(sen, startPos)
```

With the current set of error rules in GRANSKA the rule matching performs six times faster with optimization than without. Furthermore, due to the optimization, it is almost free (with respect to performance) to add many new rules as long as they contain some uncommon word or tag bigram.

7 Lexicons and Word Form Generation

The lexicon of the system and the probability estimations that are needed for the tagger are derived from the tagged Stockholm-Umeå Corpus (SUC) (Ejerhed *et al.* 1992) in addition to morphological information from various sources.

The grammar rules require the functionality to generate alternate inflection forms of any given word. Instead of having a lexicon containing all more or less common forms of each base form word, we use inflection rules to derive word forms from a base form. This approach has two advantages. Firstly, all inflectional forms of a word can be derived as long as its base form is known, and thus a smaller lexicon can be used. Secondly, unknown compound words can inherit the inflection rule of its last word form constituent, which enables corrections of unknown compound words.

8 Evaluation and Ranking of Error Corrections

It is often the case that an error rule matching generates more than one correction alternative. There are several reasons for this: different syntactic features may be applicable when a word form is changed, a base form may have more than one applicable inflection rule, and an error rule may have more than one correction field. These alternative sentences are first scrutinized and then ranked before being suggested to the user.

As the error rules are applied locally and not to an entire clause, sentence or paragraph, there will inevitably be false alarms. Therefore, each corrected sentence

generated from an error rule matching is scrutinized with all other error rules in order to determine if another error was introduced. In such cases, the correction alternative is discarded.

If one of the correction alternatives is identical to the original sentence, it indicates not that the original sentence was erroneous, but that it was incorrectly tagged. For example, the noun *verktyg* (tool) has the same spelling in singular and plural. If the tagger tags *verktyg* as a plural noun in *Ett mycket bra verktyg* (eng. A very good tool), a noun phrase disagreement error rule will correct the phrase to *Ett mycket bra verktyg*, where the only difference is the tag of the last word. Thus, when a corrected sentence identical to the original sentence is generated, the entire error matching is regarded as a false alarm.

These two approaches of discarding correction alternatives have indeed shown to increase precision more than they decrease recall.

There is another benefit from scrutinizing the sentences generated from error rules. The probability given by the tagging equation is a suitable measure for ranking these sentences, so that the sentence with most "common" words and syntactic structure is given as first alternative. We believe that it is important for a spelling and grammar checker to suggest reasonable corrections. A spelling or grammar checker that suggests a non-grammatical correction will lose in confidence from the user.[2]

If a sentence has a great proportion of unknown words, it makes little sense to apply grammar and spelling checking rules to it, since it is probably a non-Swedish sentence. Instead, such a sentence is either ignored, marked as suspect in its entirety, or scrutinized anyway, according to the user's preference.

9 Results

The tagging module has a processing speed of about 70 000 words per second on a PC with 256 MByte memory and a Pentium I with 866 MHz. In a previously unseen text, 97% of the words are correctly tagged. Unknown words are correctly tagged in 93% of the cases. The whole system (with about 20 rule categories of about 250 error rules) processes about 5 000 words per second, tagging included. The numbers are hard to compare to those of other systems, since they are seldom reported, but we believe that we have achieved a comparably high performance. High performance matters in practical applications, where the grammar checker for instance should be runned in an interactive mode, checking every new sentence written. Memory usage is also important in practical applications, and Granska consumes about 22 MB RAM.

We conducted an evaluation of Granska on a test collection comprising 200 000 words from five text genres. The text genres were sport news, international

[2] The notions of trust and credibility have received increased attention in recent research about human-computer interaction. It applies not only to language support systems, but to all systems providing information and services to a human user. A recent overview is presented in (Fogg and Tseng 1999).

	Sport news	Interna-tional news	Public authority text	Popular science	Student essays	All texts
Split compounds	100/11	–/0	71/42	60/27	40/67	46/39
Noun phrase disagreement	88/39	100/11	100/25	100/37	74/72	83/44
All error types	67/52	60/25	67/47	87/46	37/66	52/53

Table 1. *Percentages for recall/precision for two error types and all existing grammatical error types in the texts.*

news, public authority text, popular science and student essays. The test collection contained 418 syntactic errors (only 0.2 % of the number of words) of different complexity. The major error types were: verb chain errors (21%), split compounds (18%), noun phrase disagreement (17%), context-sensitive spelling errors (13%) and missing words (13%). The remaining 18% of the errors belonged to about ten broad error types. Granska tries to cover about 60% of all errors in the test collection. The overall recall on the five genres was 52% and the precision was 53%. However, there was an significant difference between the results on the different text genres, see Table 1.

9.1 Comparing Granska *to Other Grammar Checkers*

It is interesting to compare Granska to other grammar checkers with respect to grammar checking ability. However, it is hard to compare different grammar checkers for several reasons. Comparing an evaluation such as the one in Table 1 with evaluations made on other systems is difficult because of e.g. different languages, type of evaluation corpus, and error complexity. The evaluation of Granska in Table 1 seems, however, to be in line with the evaluation on the English grammar checker Critique (Richardson and Braden-Harder 1993) on different text genres.

An obvious way to compare two or more grammar checkers is to run them on the same text and compare the results. Since different grammar checkers may be specialized on different types of errors and text genres the results may vary between different evaluation texts. It is therefore important to note which text genre that was used in a comparison and which types of errors that were studied.

Lingsoft that developed the Swedish grammar checker in Microsoft Word 2000 evaluated their grammar checker on news paper texts (Birn 2000). Granskahas also been evaluated on news papers text, but not on the same corpus that Lingsoft used. However, a genre based comparision might give an indication of the performance of two grammar checker on the same text genre. Such comparison shows higher precision but lower recall for Lingsofts's grammar checker than the overall

result of GRANSKA. The Swedish error detection rules of MS Word are expressible in the rule language of GRANSKA, but not the other way around.

Sofkova Hashemi has developed a finite state based grammar checker called Finite Check (Sofkova Hashemi, Cooper, and Andersson 2003). In a comparative study of performance of four Swedish grammar checkers on texts written by primary school children (Sofkova Hashemi 2003) GRANSKA got higher F-score (23%) than both Scarrie (Sågvall Hein 1998) (18%) and the grammar checker of MS Word (Birn 2000) (14%). Finite Check showed the best results with a F-score of 49%. The comparative part of the study focused on four error types which were also the ones targeted by FiniteCheck: noun phrase agreement, finite verb form, verb form after auxiliary verb, and verb after infinitive mark. FiniteCheck was trained on the test material which partly explains the high F-score.

The low F-scores might also be explained by the fact that the evaluations of the alarms are made in a quite strict manner. An example is split compounds identified by the programs as agreement errors are counted as false alarms, in spite of the fact that they includes agreement errors, at least at a surface level, but with the most probable interpretation as a split compound. However, the study by (Sofkova Hashemi 2003) shows that there is a strong need for research and development to improve the performance of grammar checking tools for different user groups.

The rules of FiniteCheck can easily be converted into the rule language of GRANSKA, which means that GRANSKA may be improved to find the errors found by FiniteCheck.

9.2 Comparing Granska to the Spelling and Grammar Checker of Microsoft Word

We have evaluated GRANSKA and the spelling and grammar checker of Microsoft Word 2000, on texts written by second language learners of Swedish, although neither of the systems has been designed for this heterogeneous user group. The reasons that we chose this type of evaluation text were that we wanted texts containing quite a large number of real errors, and that this group of users is a growing group needing good checking tools.

The texts comprised 32 452 words and were taken from the Svante corpus (Borin 2004). The texts were written by advanced learners of Swedish as a foreign language at a Swedish university. The evaluation has focused on precision and the maximal recall of the spelling and grammar checkers. With maximal recall we mean detected errors by one program divided by detected errors by both programs. A program cannot get better recall than this maximal recall. We have chosen to include the performance of the style and spell checkers (including typographical errors) in the results. For second language learners these alarms might be as hard as grammar checking alarms to process.

The two spelling and grammar checkers have produced 787 alarms on the texts, and 362 of these alarms (1.1 % of the number of words) were true positives. The individual performance of each program is presented in Table 2. The specific results on split compounds and agreement errors are reported in Table 3. Only the numbers of error detections are measured. Error diagnosis and correction proposals are not

Error type	Max. Recall		Precision		F-score	
	Word	Granska	Word	Granska	Word	Granska
Typographical	82	21	62	70	71	31
Style	0	0	0	-	0	0
Spelling	91	57	38	86	54	68
Grammar	34	84	86	80	49	82
Total	66	63	47	82	55	71

Table 2. *Comparing the Swedish spelling and grammar checkers of* Granska *and MS Word. The figures are given in percentages for recall, precision and F-score.*

Error type	Max. Recall		Precision	
	Word	Granska	Word	Granska
Split compounds	0	100	–	81
Agreement in NP	54	74	92	100
Agreement in predicative	0	100	–	60
Total	38	81	92	86

Table 3. *Comparing the Swedish spelling and grammar checkers of* Granska *and MS Word on split compounds and agreement errors. The figures are given in percentages.*

evaluated. Especially the detections of split compounds of Granska involve some diagnosis that might be misleading for the user. However, at a surface level they can be counted as split compounds.

The grammar checker of MS Word has better precision, but much lower recall than Granska. For the spelling checkers, it is the opposite, the spelling checker of MS Word has a very high recall, but quite low precision, mostly due to bad performance on compounds, and the fact that it marks many proper names as spelling errors. The spell checker of Granska has quite a low recall; it is too liberal when judging misspelled compounds; many of them passed by undetected. Analysis of Swedish compounds still seems to be an area that needs further work, see Sjöberg and Kann (2004).

10 Conclusions

We have found that Granska is a fast gammar checker that is as good or better than comparable grammar checkers in detecting grammar errors. Furthermore, the

rule language of GRANSKA is powerful enough to implement the rules of other grammar checkers, and can therefore take advantage of good handcrafted and well tested rules of other systems. The optimized matching used in GRANSKA makes it possible to use large sets of error rules almost without increasing the processing time.

Moreover, we have implemented a Swedish chunker and surface parser using the rule language of GRANSKA (Knutsson, Bigert, and Kann 2003). This proves the rule language and the rule matching to be general enough to implement other tools for natural language analyzing of Swedish, and probably for many other languages as well. In order to port the system to another language besides new rules, just a new tagger, a spelling checker and a word form generator are needed.

It is unrealistic to hope for full recall and precision in grammar checking. Therefore, we think that it is important to develop a user friendly and instructive graphical interface and test the program on users in practice to study usability aspects as well as the effects on writing and writing ability. Two user studies which brought some light on these questions were conducted in different phases of the development of GRANSKA (Domeij, Knutsson, and Severinson-Eklundh 2002). These two studies are especially important for the next step in the development of GRANSKA, which is to adapt it to second language learners of Swedish. These users need extended error detection capacity and a more comprehensive feedback from the program. New user studies (Knutsson, Cerratto Pargman, and Severinson-Eklundh 2003) with second language learners using GRANSKA are an important guide for the directions of the future research.

The method for detection of grammar errors that we have described uses error detection rules, and will therefore find only predictable errors. We have also studied a statistical approach for finding unpredictable context-sensitive spelling errors (Bigert and Knutsson 2002) and an approach based on machine learning of errors that have been automatically inserted into a corpus (Sjöbergh and Knutsson 2005). These two approaches complement each other well, and together they find many more errors than any single approach (Bigert *et al.* 2004).

Acknowledgments

The work has been funded by the Swedish research councils TFR, HSFR, Nutek and Vinnova. Project leader of the project has been Prof. Kerstin Severinson Eklundh. We would also like to thank Johnny Bigert and Jonas Sjöbergh for their work with the current version of GRANSKA.

Språkdata at Göteborg University and the Swedish Academy let us use Svenska Akademiens ordlista as a source for words in GRANSKA. Prof. Eva Ejerhed and Prof. Gunnel Källgren let us use SUC.

References

A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques and Tools.* Addison-Wesley, Reading, Mass., 1986.

A. Arppe. Developing a grammar checker for Swedish. In *Proc. 12th Nordic Conf. in Computational Linguistics (Nodalida-99)*, pages 13–27, 2000.

E. S. Atwell. How to detect grammatical errors in a text without parsing it. In *Proc. 3rd EACL, Copenhagen, Denmark*, pages 38–45, 1987.

K. R. Beesley and L. Karttunen. *Finite State Morphology.* CSLI Publications, Stanford, CA, 2003.

J. Bigert, V. Kann, O. Knutsson, and J. Sjöbergh. Grammar checking for Swedish second language learners. In P. J. Henrichsen, editor, *CALL for the Nordic Languages*, Copenhagen Language Studies, Samfundslitteratur, Copenhagen, Denmark. 2004.

J. Bigert and O. Knutsson. Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proc. 2nd Workshop Robust Methods in Analysis of Natural language Data (ROMAND'02), Frascati, Italy*, pages 10–19, 2002.

J. Birn. Detecting grammar errors with Lingsoft's Swedish grammar checker. In *Proc. 12th Nordic Conf. in Computational Linguistics (Nodalida-99)*, pages 28–40, 2000.

L. Borin. The SVANTE project's home page, 2004. `http://svenska.gu.se/~svelb/svante/`.

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

A. Carlson, J. Rosen, and D. Roth. Scaling up context sensitive text correction. In *Proc. 13th Nat. Conf. Innovative Applications of Artificial Intelligence (IAAI'01)*, 2001.

E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts, 1996.

E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In *Proc. 11th Nat. Conf. Artificial Intelligence*, pages 784–789, 1993.

R. Domeij, J. Hollman, and V. Kann. Detection of spelling errors in Swedish not using a word list en clair. *J. Quantitative Linguistics*, 1:195–201, 1994.

R. Domeij, O. Knutsson, and L. Öhrman. Inkongruens och felaktigt särskrivna sammansättningar – en beskrivning av två feltyper och möjligheten att detektera felen automatiskt (Incongruence and erroneously split compounds), in Swedish. In *Proc. Svenskans beskrivning-99*, 1999.

R. Domeij, O. Knutsson, and K. Severinson-Eklundh. Different ways of evaluating a Swedish grammar checker. In *Proc. 3rd Int. Conf. Language Resources and Evaluation (LREC 2002), Las Palmas, Spain*, 2002.

E. Ejerhed. Finite state segmentation of discourse into clauses. In A. Kornai, editor, *Extended Finite State Models of Language*, chapter 13. Cambridge University Press, 1999.

E. Ejerhed, G. Källgren, O. Wennstedt, and M. Åström. The linguistic annotation system of the Stockholm-Umeå corpus project. Technical Report DGL-UUM-R-33, Department of General Linguistics, University of Umeå, Umeå, 1992. The web page of SUC is `www.ling.su.se/DaLi/Projects/SUC/`.

B. J. Fogg and H. Tseng. The elements of computer credibility. In *Proc. Human Factors in Computing Systems (CHI-99)*, pages 80–87, Pittsburgh, PA, 1999. ACM Press.

A. R. Golding and D. Roth. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1–3):107–130, 1999.

G. E. Heidorn. Intelligent writing assistance. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*, chapter 8, pages 181–207. Marcel Dekker, New York, 2000.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. Automatic error detection in the japanese learners english spoken data. In *Companion Volume to Proc. ACL'03, Sapporo, Japan*, pages 145–148, 2003.

K. Jensen, G. Heidorn, and S. Richardson. *Natural Language Processing: The PLNLP Approach.* Kluwer, Boston, 1993.

K. Jensen, G. E. Heidorn, L. A. Miller, and Y. Ravin. Parse fitting and prose fixing: Getting a hold on ill-formedness. *Comp. Linguistics*, 9(3-4):147–160, 1983.

V. Kann, R. Domeij, J. Hollman, and M. Tillenius. Implementation aspects and applications of a spelling correction algorithm. In L. Uhlirova, G. Wimmer, G. Altmann, and R. Koehler, editors, *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, volume 60 of *Quantitative Linguistics*, pages 108–123. WVT, Trier, Germany, 2001. Available at `http://www.nada.kth.se/theory/projects/swedish.html`.

F. Karlsson. Constraint Grammar as a framework for parsing running text. In H. Karlgren, editor, *Proc. 12th Int. Conf. Computational Linguistics (COLING-90)*, volume 3, pages 168–173, 1990.

O. Knutsson. *Automatisk språkgranskning av svensk text (Automatic Proofreading of Swedish Texts), in Swedish.* Licentiate thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Sweden, 2001.

O. Knutsson, J. Bigert, and V. Kann. A robust shallow parser for Swedish. In *Proc. 14th Nordic Conf. on Computational Linguistics*, 2003.

O. Knutsson, T. Cerratto Pargman, and K. Severinson-Eklundh. Transforming grammar checking technology into a learning environment for second language writing. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 38–45, 2003.

K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

L. Mangu and E. Brill. Automatic rule acquisition for spelling correction. In *Proc. 14th International Conference on Machine Learning*, pages 187–194, 1997.

R. T. Martins, R. Hasegawa, M. Das Graças VolpeNunes, G. Montilha, and O. N. De Oliveira, Jr. Linguistic issues in the development of regra: A grammar checker for Brazilian Portuguese. *Natural Language Engineering*, 4(4):287–307, 1998.

O. Wedbjer Rambell. Error typology for automatic proof-reading purposes. Technical Report Scarrie del. 2.1, final version 1.1, Department of Linguistics, Uppsala University, Uppsala, Sweden, 1999.
Available at `http://stp.ling.uu.se/~matsd/thesis/arch/2000-009.pdf`.

S. Richardson and L. Braden-Harder. The experience of developing a large-scale natural processing system: Critique. In K. Jensen, G. E. Heidorn, and S. D. Richardson, editors, *Natural Language Processing: The PLNLP Approach*, pages 77–89. Kluwer, Boston, 1993.

A. Sågvall Hein. A chart-based framework for grammar checking. In *Proc. 11th Nordic Conf. in Computational Linguistics (Nodalida-98)*, 1998.

J. Sjöbergh and V. Kann. Finding the correct interpretation of Swedish compounds, a statistical approach. In *Proc. 4th Int. Conf. Language Resources and Evaluation (LREC 2004)*, 2004.

J. Sjöbergh and O. Knutsson. Faking errors to avoid making errors: Very weakly supervised learning for error detection in writing. In *Proc. RANLP 2005*, 2005.

S. Sofkova Hashemi. *Automatic Detection of Grammar Errors in Primary School Childrern's texts.* PhD thesis, Department of Linguistics, Göteborg University, Sweden, 2003.

S. Sofkova Hashemi, R. Cooper, and R. Andersson. Positive grammar checking: A finite state approach. In *Computational Linguistics and Intelligent Text Processing. 4th International Conference, CICLing2003*, 2003.

T. Vosse. *The Word Connection. Grammar-Based Spelling Error Correction in Dutch.* Enschede: Neslia Paniculata, 1994. ISBN 90-75296-01-0.