Granska
- an efficient hybrid system for Swedish grammar checking

Rickard Domeij, Ola Knutsson, Johan Carlberger, Viggo Kann
Nada, KTH, Stockholm
Dept. of Linguistics, Stockholm University
{domeij, knutsson, jfc, viggo}@nada.kth.se

***

Abstract

This article describes how Granska – a surface-oriented system for checking Swedish grammar – is constructed. With the use of special error detection rules, the system can detect and suggest corrections for a number of grammatical errors in Swedish texts. Specifically, we focus on how erroneously split compounds and noun phrase agreement are handled in the rules.

The system combines probabilistic and rule-based methods to achieve high efficiency and robustness. This is a necessary prerequisite for a grammar checker that will be used in real time in direct interaction with users. We hope to show that the Granska system with higher efficiency can achieve the same or better results than systems that use rule-based parsing alone.

1. Introduction

Grammar checking is one of the most widely used tools within language technology. Spelling, grammar and style checking for English has been an integrated part of common word processors for some years now. For smaller languages, such as Swedish, advanced tools have been lacking. Recently, however, a grammar checker for Swedish has been launched in Word 2000 and also as a stand-alone system called Grammatifix (Arppe 2000, this volume; Birn 2000, this volume).

There are many reasons for further research and development of grammar checking for Swedish. First, the need for writing aids has increased, both concerning the need for more efficiency and quality in writing. Secondly, the linguistic analysis in grammar checking needs further development, especially in dealing with special features in Swedish grammar and its grammatical deviations. This is a development that most NLP-systems will benefit from, since they often lack necessary methods for handling ungrammatical input. Thirdly,

Proceedings of NODALIDA 1999, pages 49-56

there is need for more sophisticated methods for evaluating the functionality and usability of grammar checkers and their effect on writing and writing ability.

There are two research projects that focus on grammar checking for Swedish. These projects have resulted in two prototype systems: Scarrie (Sågvall-Hein 1998; Scarrie 2000) and Granska (Domeij, Eklundh, Knutsson, Larsson & Rex 1998). In this article we describe how the Granska system is constructed and how grammatical errors are handled by its error rule component. The focus will be on the treatment of agreement and split compound errors, two types of errors that frequently occur in Swedish texts.

2. The Granska system

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. This is a necessary prerequisite for a grammar checker that runs in real time in direct interaction with users (e.g. Kukich 1992). Using special error rules, the system can detect a number of Swedish grammar problems and suggest corrections for them.

In figure 1 the modular structure of the system 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 part of speech and inflectional form information to each word. The tagged text is then sent to the error rule 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. Furthermore, the system contains a spelling detection and correction module which can handle Swedish compounds (Kann, Domeij, Hollman & Tillenius 1998). The spelling detection module can be used from the error rules for checking split compound errors.

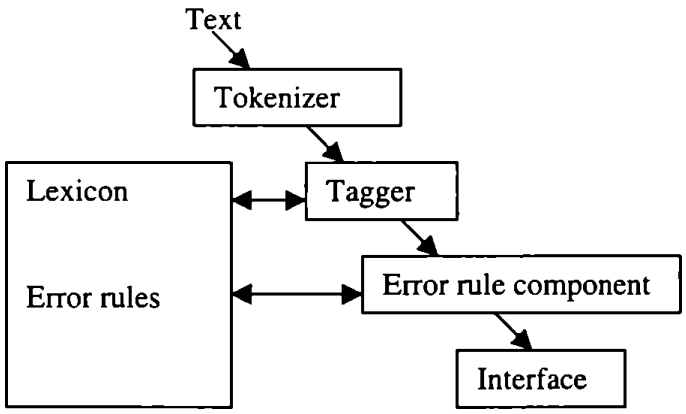

Figure 1. *An overview of the Granska system.*

The system is implemented in C++ under Unix and there is also a web site where it can be tested from a simple web interface (see www.nada.kth.se/theory/projects/granska/demo.html). There is ongoing work for designing a graphical interface for PC which can be used interactively during writing. The PC system will be used as a research tool for studying usability aspects with real users.

3. Tagging and lexicon

The Granska system uses a hidden Markov model (Carlberger & Kann 1999) to tag and disambiguate all words in the input text. Every word is given a tag that describes its part of speech and morphological features. The tagging is done on the basis of a lexicon with 160 000 word forms constructed from SUC, a hand tagged corpus of one million words (Ejerhed, Källgren, Wennstedt & Åström 1992). The lexicon has been further complemented with words from SAOL, the Swedish Academy's wordlist (Svenska akademien 1986). The Markov model is based on statistics from SUC about the occurrence of words and tags in context. From this information the tagger can choose the most probable tag for every word in the text if it is listed in the lexicon. Unknown words are tagged on the basis of probabilistic analysis of word endings.

4. Error rules

The error rule component uses special 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 ***ett röd bil*** (a red car), where *en* (a) does not agree with *röd* (red) and *bil* (car) 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 (e.g. Sågvall-Hein 1998). An error rule in Granska that can detect the agreement error in ***ett röd bil*** is shown in rule 1 below.

Rule 1:

```
kong22@inkongruens
{
X(wordcl=dt),
Y(wordcl=jj)*,
Z(wordcl=nn & (gender!=X.gender | num!=X.num | spec!=X.spec))
-->
mark(X Y Z)
corr(X.get_form(gender:=Z.gender, num:=Z.num, spec:=Z.spec) Y Z)
info("Artikeln" X.text "stämmer inte överens med substantivet" Z.text)
action(granskning)
}
```

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.

More formally, the condition part of the rule can be read as "an X with the word class determiner (i.e. wordcl=dt) followed by zero or more Y:s with the word class adjective (i.e. wordcl=jj*) and a Z with the word class noun (i.e. wordcl=nn) for which the values of gender, number or species are not agreeing with the corresponding values of the determiner X (i.e. gender!=X.gender | num!=X.num | spec!=X.spec). The characters "=","!=", "|" and "&" denotes the operators "is identical to", "is not identical to", "or" and "and" respectively. The comma is used for separating matching variables. The Kleene star (*) indicates that the preceding object can have zero or more instances.

Examples of phrases that match the condition is ***ett*** *röd bil* (deviation in gender), ***en*** *röda bilen* (deviation in species) and ***den*** *röda bilarna* (deviation in number).

The action part of the rule specifies in the first line after the arrow that the erroneous phrase X Y Z should be marked in the text. In the second line of the action part, a function (X.get_form) is used to generate a new inflection of the article X from the lexicon, one that agrees with the noun Z. When calling this function, the determiner X is assigned the same values of gender, number and species as the noun Z by the operator ":=" in order to get a new form from the lexicon that agrees with the noun. The new form is presented to the user as a correction suggestion (in the example ***en*** *röd bil*) by the corr statement. In the info statement in line 3, a diagnostic comment describing the error is constructed and presented to the user.

In most cases, the tagger succeeds in choosing the correct tag for the deviant word on probabilistic grounds (in the example *ett* is correctly analyzed as an indefinite, singular and neuter determiner by the tagger). However, since errors are statistically rare compared to grammatical constructions, the tagger can sometimes choose the wrong tag for a morpho-syntactically deviant form. In such cases, when the tagger is known to make mistakes, the error rules can be used in retagging the sentence to correct the tagging mistake. Thus, a combination of probabilistic and rule-based methods is used even during basic word disambiguation.

5. Help rules

It is possible to define phrase types like noun phrase (NP) and prepositional phrase (PP) in special help rules that can be used from any error rule. Rule 2 below, uses two help rules as subroutines (NP@ and PP@) in detecting agreement errors in predicative position. The help rules specify the internal structure of the NP and the PP in the main rule (pred2@predikativ). Note that the help rule PP@ uses the other help rule NP@ to define the prepositional phrase.

The main rule states that the copula X should be preceded by an NP optionally followed by zero or more PPs, and that an adjective Y that does not agree with the NP in gender or number should follow the copula. An example of a sentence matching the rule is *det lilla huset vid sjön är röd* (the little house by the lake is red) where the form *röd* does not agree in gender with the NP. The variables T and Z in the rule are contextual variables that

ensures that the NP is not part of a previous prepositional phrase and that the adjective Y in the supposed predicative position is not part of a larger noun phrase.

Rule 2:

```
pred2@predikativ
{
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(*)
corr(T NP PP X Y.get_form(gender:=NP.gender, num:=NP.num, spec:=ind) Z)
info("Substantivfrasen" NP.text "stämmer inte överens med
"adjektivet" Y.text)
action(granskning)
}


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


PP@
{
X(wordcl=pp),
(NP)
-->
action(hjälp)
}
```

The help rules make the analysis approaches that of a phrase structure grammar. Help rules make it possible for the system to do 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.

6. Erroneously split compounds

Above we have shown how agreement errors are handled in the system. Another frequently occurring error type is erroneously split compounds. In contrast to English, a Swedish compound is regularly formed as one word so split compounds are treated as

ungrammatical. So far, we have mainly focussed on erroneously split compounds of the type noun+noun which stands for about 70 % of the various types (Domeij, Knutsson & Öhrman 1999).

Detection of erroneously split compounds where the first part cannot stand alone is trivial. This 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. An example is *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 uninflected noun followed by any noun is an unusual syntactical combination in grammatically correct sentences, it can be used to find candidates for split compound errors. Other contextual cues are also used before checking the candidate against a spell checker for compound recognition. If the spell 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.

Rule 3.

```
sär2@särskrivning
(
X1(wordcl=dt),
X2(wordcl=jj)*,
X3(wordcl=nn & (gender!=X1.gender | num!=X1.num | spec!=X1.spec)),
X4(wordcl=nn & gender=X1.gender & num=X1.num & spec=X1.spec &
correctly_spelled(concat(X3.text, X4.text)))
-->
mark(X3 X4)
corr(X1 X2 concat(X3.text, X4.text))
info("Särskrivningsfel, sätt ihop" X3 X4  "till " concat(X3.text, X4.text))
action(granskning)
)
```

In rule 3 above (which has been slightly simplified), an erroneously split compound is described where the determiner X1 does not agree with the first noun X3, but does agree with the second noun X4 as in the phrase *ett cykel ställ* (a bike rack). If the two nouns were to be combined into one word the result would be a perfectly grammatical phrase (*ett cykelställ*). Therefore, the disagreement between determiner and the following noun, together with the agreement between determiner and second noun give reasonable contextual evidence to suspect an erroneously split compound. To corroborate this hypothesis further, the two nouns are combined and checked by the spell checking function, as indicated in the last line of the condition, to see if the combined words are recognized as a compound. In the action part of rule 3, the error candidate is first marked in the text and then concatenated to be used in the error correction suggestion.

It can happen that two matching error rules collide, as in the example *ett cykel ställ* where both the rule for agreement error and the rule for erroneously split compound apply in an overlapping fashion. At the time, there is nothing to prevent the system to interpret

this error in both ways. However, the problem can sometimes be avoided by further disambiguation in the rules. For harder cases, a possibility would be to order the rules corresponding to the probability that an error occurs in a given context. Before we make a decision to implement such a function, we need to look deeper into the problem. Often, the wisest strategy is to present all error possibilities to the user.

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 erroneously 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.

There are also problems with false alarms, for example when the tagger has mistagged a word so that the error rules make a false detection.

7. Results

The tagging module has a processing speed of more than 22 000 words per second on a SUN Sparc station Ultra 5. In a previously unseen text, 97 percent of the words are correctly tagged, a good result in an international comparison. Words not covered by the dictionary are correctly tagged in 92 percent of the cases. The whole system (with about 20 rule types for 250 error rules) processes about 2 800 words per second, tagging included. The numbers are hard to compare since results for other systems are seldom sufficiently presented, but we believe that we have achieved a comparably high performance. The results show that by using statistical methods it is possible to achieve a reasonably good linguistic analysis combined with high efficiency for real-time computation. We also believe that the analysis can be further improved by using rule-based methods for correcting faulty statistical analysis.

We are still working with optimizing the system and improving the error rules. Preliminary tests with the error rules show that we can hope for a recall rate above 50 percent and a precision rate above 90 percent for agreement errors and erroneously split compounds. The results so far are promising, but we need further development and testing before we present final results and compare them to fully developed systems, such as Grammatifix (Arppe, 2000, this volume; Birn 2000, this volume). Even if Granska is not yet fully developed, it has the advantage of being able to detect split compounds in Swedish, something that neither Grammatifix or any other commercial system does.

It is unrealistic to hope for full recall and precision. Therefore, we think it is important to test the program on users in practice to study usability aspects as well as the effects on writing and writing ability (see Domeij 1997, 1998). To do that we need to develop a user friendly and instructive graphical interface. The graphical interface for PC is scheduled to be ready during the spring 2000. The user tests will be ready before the end of the year.

Acknowledgements

The work has been funded by the Swedish research councils TFR, HSFR and Nutek. Project leader has been Kerstin Severinson-Eklundh. Språkdata at Göteborg University and the Swedish Academy let us use Svenska Akademiens ordlista as a source for words in Granska. Prof. Eva Ejerhed from Umeå University and Prof. Gunnel Källgren from Stockholm University let us use SUC.

References

Arppe, A. 2000. *Developing a Grammar Checker for Swedish*, Nodalida'99, Trondheim, December 1999.

Birn, J. 2000. *Detecting Grammar Errors with Lingsoft's Swedish Grammar Checker.* Nodalida'99, Trondheim, december 1999.

Carlberger, J. & Kann, V. 1999. Implementing an Efficient Part-of-Speech Tagger. In: *Software - Practice and Experience*, 29 (9), pp. 815-832.

Domeij, R. 1997. Datorn och språkriktigheten. I: O. Josephson (ed.) *Svenskan och IT-samhället.* Uppsala: Hallgren & Fallgren.

Domeij, R., Eklundh, K., Knutsson, O., Larsson, S. & Rex, Å. (1998). Granskaprojektet 1996-1997. Technical Report NADA, KTH.

Domeij, R. 1998. Detecting, Diagnosing and Correcting Low-Level Problems when Editing with and without Computer Aids. In *TEXT Technology*, vol 8, no. 1. Wright State University, Celina, USA.

Domeij, R., Knutsson, O. & Öhrman, L. 1999. *Inkongruens och felaktigt särskrivna sammansättningar - en beskrivning av två feltyper och möjligheten att detektera felen automatiskt.* Svenskans beskrivning, October 1999.

Ejerhed, E., Källgren, G., Wennstedt, O. & Åström, M. 1992. The Linguistic Annotation System of the Stockholm-Umeå Corpus Project. Description and Guidelines. Version 4.31. Department of Linguistics, Umeå University.

Kann, V., Domeij, R., Hollman, J., & Tillenius, M. 1998. Implementaion Aspects and Applications of a Spelling Correction Algorithm. To appear in: R. Koehler, L. Uhlirova, G. Wimmer: *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, 1999. NADA report TRITA-NA-9813, 1998.

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

Scarrie. 2000. Web site: stp.ling.uu.se/~ljo/scarrie-pub/scarrie.html

Svenska akademien (The Swedish Academy) 1986. Ordlista över det svenska språket (SAOL), 11th edition. Stockholm: Norstedts Förlag.

Sågvall-Hein, A. 1998. A Chart-Based Framework for Grammar Checking. *Proc. from Nodalida98.*