A Swedish Grammar Checker

Johan Carlberger Rickard Domeij Viggo Kann Ola Knutsson*
Royal Institute of Technology, Stockholm

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 noun phrase 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 hope to show 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 engineering. Spelling, grammar and style checking for English has been an integrated part of common word processors for some years now. However, most such programs are strictly commercial, and therefore there exists no complete documentation of algorithms and rules used. A good exception is the rule-based system by Vosse (1994). Current research seem to focus on so-called sensitive spell checking (see for example (Golding and Roth, 1999; Carlson, Rosen, and Roth, 2001)).

For smaller languages, 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 based on the Swedish constraint grammar SWECG. There are also two research prototypes available for Swedish, Scarrie (Sågvall Hein, 1998) and a system using a finite state approach (Hashemi, 2001).

In this article, another grammar checker for Swedish is presented. This grammar checker, called GRANSKA, has been developed at KTH for about four years. 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 the tagging and error detection.

The method we describe here uses error detection rules, and will therefore find only predictable errors. We have also studied another approach for finding unpredictable context-sensitive spelling errors (Bigert and Knutsson, 2002).

* Nada, Department of Numerical Analysis and Computer Science, KTH, SE-100 44 Stockholm.
E-mail: {jfc,domeij,viggo,knutsson}@nada.kth.se.

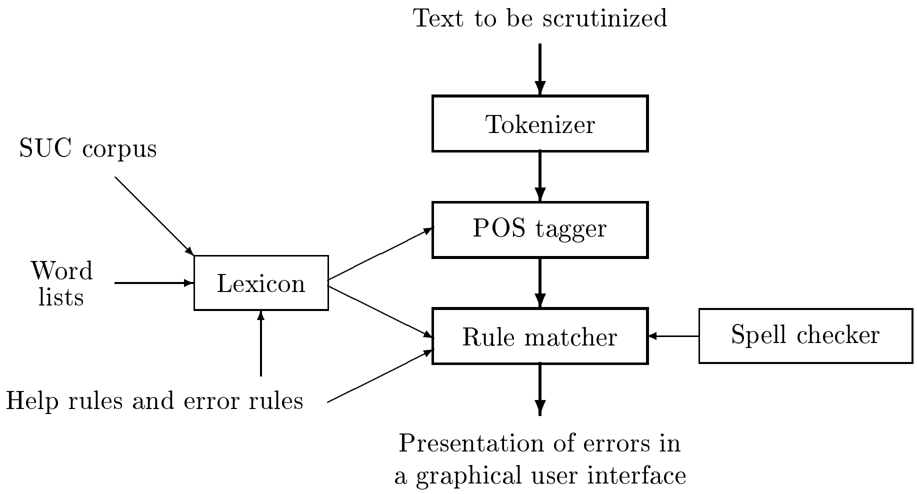

Figure 1
An overview of the GRANSKA system.

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, which is a necessary prerequisite for a grammar checker that runs in real time in direct interaction with users (Kukich, 1992). Using so called 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 disambiguated part of speech and inflectional form information to each word (Carlberger and Kann, 1999). The tagged text is then sent to surface grammar help rules that find structures such as noun phrases and then 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. Finally, the system contains a spelling detection and correction module that can handle Swedish compounds (Domeij, Hollman, and Kann, 1994; Kann et al., 2001). Incorporating the spelling detection and correction in GRANSKA improves its performance. For example it does not need to detect unknown proper names as spelling errors, and corrections that do not fit in the context are not proposed. The spelling detection module can also be used from inside the error rules, e.g. for checking split compound errors.

The GRANSKA system is implemented in C++ under UNIX, and it can be tested using a simple web interface[1]. The system is used as a research tool for studying usability aspects with real users.

1 See the web page of the project: `http://www.nada.kth.se/theory/projects/granska/`

3 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

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

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 98% of known and 93% of unknown words correctly using a tag set of size 140 (Carlberger and Kann, 1999). We have found that nearly all tags in the tag set is 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 non-grammatical text be tagged? For example, should *red* in *a red apples* be tagged as singular or plural, or as both? We have found that it is almost always better to choose one of the taggings, since if *red* is tagged as singular then the error rules will detect *red apples* as an incongruence, and if *red* is tagged as plural then *a red* will be detected as an incongruence. Thus it is better to disambiguate even when it is not clear how to do it.

4 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; Sågvall Hein, 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 Java expression 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* (sub-routines) 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.

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.

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 regarded 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, and 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, 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 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.

We also hope that help rules for clause boundary recognition could even more increase recall and precision, 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, GRANSKA can avoid false alarms and still detect errors in another clause within the same sentence.

5 Rule Matching

Presently, there are about 180 scrutinizing rules, 60 help rules and 110 accepting rules (rules that sort out exceptions) in Granska. This could be compared to Word's grammar checker 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. Since the structure of the rules is in practice not very complicated a simple recursive matching algorithm is used and works well. 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.

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. 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. The program itself detects the optimal positions in each rule and stores two tables representing this information on disk. The first table describes, for each tag bigram, which rules that should be checked when that tag bigram occurs in the text. The second table 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.

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.

6 Lexicons and Word Form Generation

The lexicon of the system is derived from the tagged Stockholm-Umeå Corpus (SUC) 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.

7 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* (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 spell and grammar checker to suggest reasonable corrections. A spell or grammar checker that suggests a non-grammatical correction will lose in confidence from the user. 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).

If a sentence has a great proportion of unknown words, it makes little sense to apply grammar and spell 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.

8 Results

	Sport news	International 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 error types.

The tagging module has a processing speed of more than 20 000 words per second on a SUN Sparc station Ultra 5. In a previously unseen text, 97% of the words are correctly tagged, a good result in an international comparing. 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 3 500 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.

An evaluation of Granska was conducted on a test collection comprising 200 000 words from five text genres. The text genres were sport news, international news, public authority text, popular science and student essays. The test collection contained 418 syntactic errors 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.

Comparing the evaluation with evaluations made on other systems is difficult because of different languages, type of evaluation corpus, error complexity and more. However, the evaluation of Granska seems to be in line with the evaluation on the English grammar checker Critique (Richardson and Braden-Harder, 1993) on different text genres. An evaluation on Word's grammar checker (Birn, 2000) showed higher precision but lower recall than the overall result of Granska on news paper texts. Comparisons on the same news paper text and other text genres still remain to be done. One notable difference is that Word's grammar checker does not address the complex error type split compound, which Granska does with some loss of precision as a result.

It is unrealistic to hope for full recall and precision. Therefore, we think 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 with second language learners using Granska will be an important guide for the directions of the future research.

Acknowledgments

The work has been funded by the Swedish research councils TFR, HSFR and Nutek. Project leader of the project has been Prof. 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 and Prof. Gunnel Källgren let us use SUC.
We would also like to thank the anonymous reviewers.

References

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

Bigert, J. and O. Knutsson. 2002. 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.*

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

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

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

Charniak, E. 1996. *Statistical language learning*. MIT Press, Cambridge, Massachusetts.

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

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

Domeij, R., O. Knutsson, and L. Öhrman. 1999. 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*.

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

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

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

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

Hashemi, S. S. 2001. Detecting grammar errors in children's writing: A finite state approach. In *Proc. 13th Nordic Conf. in Computational Linguistics (Nodalida-01)*.

Kann, V., R. Domeij, J. Hollman, and M. Tillenius. 2001. 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*. WVT, Trier, Germany, pages 108–123. Available on WWW from `www.nada.kth.se/theory/projects/swedish.html`.

Knutsson, O. 2001. *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.

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

Richardson, S. and L. Braden-Harder. 1993. 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*. Kluwer, Boston, pages 77–89.

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

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