
Nada är en gemensam institution mellan
Kungliga Tekniska högskolan och Stockholms universitet.

# Developing and Evaluating Language Tools for Writers and Learners of Swedish

OLA KNUTSSON

Avhandling som med tillstånd av Kungliga Tekniska högskolan
framlägges till offentlig granskning för avläggande av filosofie doktorsexamen
måndagen den 17 oktober 2005 kl 14.15
i sal D2, D-huset, Lindstedtsvägen 5,
Kungliga Tekniska högskolan, Stockholm.

TRITA-NA-0530
ISSN 0348-2952
ISRN KTH/NA/R--05/30--SE
ISBN 91-7178-157-9

## Abstract

Writing and written language play today an increasingly important part in many people's lives. Written language has become more or less a prerequisite for daily communication. This development of society leads to increased needs for tools that can help humans in dealing with text. A technology that has a potential to aid people with writing and written language is language technology. In this thesis, the focus is on language tools based on language technology that can aid writers and learners of Swedish.

A language tool that has been developed and evaluated in the thesis is the grammar checker Granska. The thesis work on Granska includes the design of its rule language, and the development of grammar checking rules for common error types in Swedish. In addition, rules for phrase analysis and clause boundary detection have been developed constituting a partial and shallow parser called GTA.

Language tools for writing can mainly be evaluated in two ways: with focus on text or with focus on the writer. In this thesis, both types of evaluations have been carried out both with native writers and second language writers. The first textual evaluation of Granska showed that the genre has a strong influence on the result. In a second evaluation, Granska was compared with a commercial grammar checker on second language writers' texts. Granska found more errors, but with a lower precision. A third evaluation focused on the general text analyzers, which Granska relies on, in this case a statistical word class analyzer and the parser GTA. These programs were evaluated on texts where spelling errors were introduced, in order to test the programs' robustness. Results showed that as long as the word class analyzer is robust the parser GTA would also be robust. In a first formative user study with Granska and five participants, results suggested that several and competing error diagnoses and correction proposals are not a problem for the users as long as there exist at least one accurate correction proposal. Moreover, false alarms from the spelling checker seemed to pose a limited problem for the users, but false alarms on more complicated error types might disturb the revision process of the users.

In order to improve the design of language tools for second language writers a field study was carried out at a Swedish university. Sixteen students with different linguistic and cultural backgrounds participated in the study. The objective was to study the use of Granska in students' free writing. The results indicated that although most alarms from Granska are accurate, lack of feedback and misleading feedback are problems for second language writers. The results also suggested that providing the students with feedback on different aspects of their interlanguage, not only errors, and facilitating the processes of language exploration and reflection are important processes to be supported in second-language learning environments. These insights were used as design principles in the design and development of an interactive language environment called Grim. This program includes a basic word processor, in which the user can get feedback on linguistic code features from different language tools such as Granska and GTA. In addition, other tools are available for the user to explore language use in authentic texts and to achieve lexical comprehension through bilingual dictionaries.

**Sammanfattning**

Skrivande och skrivet språk är idag en viktig del av många människors liv, i datorns ordbehandlare, i e-postprogram och i chattkanaler på Internet. Skrivet språk har blivit mer eller mindre en förutsättning för människors dagliga kommunikation. Denna utveckling av samhället leder till ökade behov av att på olika sätt hantera text. En teknologi som har stor potential att hjälpa människor med skrivande och skrivet språk är språkteknologi. I denna avhandling ligger fokus på olika språkverktyg vars avsikt är att stödja skribenter och de som lär sig svenska bland annat genom att skriva.

Ett språkverktyg som har utvecklats och utvärderats i avhandlingen är språkgranskningsverktyget Granska. I arbetet med Granska har fokus legat på utvecklingen av regelspråk, granskningsregler och generella analysregler samt utvärdering av dessa. Granska kombinerar en statistisk grundanalys av ordens ordklasser med regelbaserade metoder för sökning av grammatiska fel och frasanalys. I utvecklingen av granskningsreglerna är dragkampen mellan felaktiga utpekningar av fel, så kallade falska alarm, och uteblivna utpekningar av fel, det största enskilda problemet. Dragkampen uppstår genom att det är svårt att hitta många fel utan att också göra en del felaktiga utpekningar.

Språkverktyg för skrivande kan i stort sett utvärderas på två sätt: med fokus på texten eller på den som skriver. I denna avhandling har båda typerna av utvärdering utförts med såväl modersmålskribenter som skribenter med svenska som andraspråk. I en första textbaserad utvärdering visade det sig att textgenre spelar stor roll för Granskas resultat. Ett vanligt fel i en textgenre förekommer nästan inte alls i en annan. Detta innebär att det blir mycket svårt för programmet att inte avge några falska alarm i de texter där feltypen saknas. I en andra textbaserad utvärdering jämfördes Granska och en kommersiell grammatikkontroll på texter från andraspråksskribenter. Den kommersiella grammatikkontrollen visade sig att ha bättre träffsäkerhet, men upptäckte färre fel än Granska.

En första mindre användarstudie utfördes med Granska och fem erfarna skribenter. Syfte med studien var att utveckla Granska i linje med skribenters behov vid revision av text. Resultatet indikerade att användarna inte hade några problem med att välja mellan olika feldiagnoser om ett av ersättningsförslagen var korrekt. Falska alarm verkade vara av varierande svårighetsgrad: falska alarm från stavningskontrollen är mer eller mindre ofarliga, medan falska alarm från

granskningen av mer komplicerade feltyper kan försvåra revisionsarbetet för användaren.

Granska utvecklades från början för erfarna skribenter med svenska som modersmål, men allteftersom arbetet har fortskridit har även skribenter med svenska som andraspråk blivit en allt viktigare användargrupp. I detta arbete har diskussionen om granskningsmetod blivit mer och mer central. Även om gruppen andraspråksskribenter är mycket heterogen, så innehåller den här gruppens texter generellt sett mer fel, och i många fall fler fel i samma mening. Detta gör granskningsproblemet betydligt svårare. För det första så blir det svårare att avgöra ordens ordklass och frastillhörighet när flera fel finns i samma mening, och därmed har programmet allt mindre att hänga upp den grundläggande språkliga analysen på. För det andra är det svårare att konstruera granskningsregler för fel vars natur är svår att förutsäga på förhand.

För att förbättra den grundläggande språkanalysen utvecklades programmet GTA, som gör en frasanalys och satsgränsigenkänning. GTA utvecklades ur de generella analysregler som redan fanns i Granska. GTA designades för att klara av att analysera texter som innehåller vissa avvikelser från språkets norm, t.ex. inkongruens. För att ta reda på hur väl programmet klarade av mindre avvikelser i form av stavfel utvärderades GTA och även två program för ordklassanalys på texter med olika andel stavfel. GTA bygger till mycket stor del på att identifikationen av ordklass fungerar för att fraser och satsgränser skall analyseras korrekt. Detta bekräftas också i utvärderingen, där det visade sig att GTA klarar sig bra så länge som den underliggande ordklassanalysen klarar att hantera avvikelser i texten. En viktig faktor för att klara språkliga avvikelser, i form av stavfel, är en fungerande metod för att hantera ord som är okända för programmet.

Nya metoder för språkgranskning har undersökts i samarbete med andra forskare, och där har avhandlingens bidrag varit i form av transformationsregler i den statistiska språkgranskaren ProbGranska. Dessa regler visade sig vid en utvärdering avsevärt förbättra ProbGranskas säkerhet när det gällde att identifiera grammatiska problem. I utvecklingen av språkgranskaren SnålGranska har avhandlingen bidragit med idéer till dess grundläggande algoritm. Denna algoritm bygger på att träna ett maskininlärningsprogram på konstgjorda fel i avsaknad av en korpus med många uppmärkta autentiska fel.

För att komma vidare med utvecklingen av språkverktyg för andraspråksskribenter genomfördes en längre fältstudie vid ett svenskt

universitet. Syftet var att studera användningen av Granska i autentiska skrivuppgifter som studenterna genomförde i en avancerad kurs i svenska som främmande språk. Sexton studenter med olika språklig och kulturell bakgrund deltog i studien. En viktig del av studien utgjordes av studenternas bedömningar av Granskas alarm. Bedömningarna gjordes på en betygsskala från 1 till 5. Studenternas texter samlades också in i två versioner; en version före och en efter användningen av programmet. Denna metod gjorde det möjligt att studera i vilken grad studenterna följde Granskas råd, och huruvida dåliga eller bra råd från programmet fick höga eller låga betyg. Mest alarmerande var att dåliga råd angående ordföljd alltid fick högsta betyg. Andra ofta lämpliga råd dömdes ut för att beskrivningen av dessa feltyper, t.ex. anmärkningar om saknade tempusböjda verb och uteblivna subjekt, var svåra att förstå samt att de saknade ersättningsförslag.

En viktig insikt från fältstudien var att Granska eller liknade verktyg inte är det enda verktyg som andraspråksskribenter behöver när de skriver text. Denna insikt tillsammans med andra resultat från fältstudien mynnade ut i flera designprinciper för program med fokus på andraspråksskribenter. Dessa designprinciper användes sedan i utformningen av språkmiljön Grim. Grim är en ordbehandlingsmiljö med olika interaktiva språkverktyg integrerade: Granska, GTA, den statistiska språkgranskaren ProbGranska, lexikonet Lexin med åtta olika språkpar, konkordansgränssnitt mot stora textmängder från korpusen Parole, och en ordböjningsfunktion. I Grim kan användaren arbeta med egna eller andras texter, och få återkoppling på språkets former från Granska och GTA, undersöka ords användning i autentiska texter samt få en ökad förståelse av målspråket genom integrerade tvåspråkiga lexikon.

**Acknowledgements**

During my thesis work I have been working in a very nice and stimulating research group. What I have not told these people is how much pleasure I have had together with them. Some people might think that research project meetings are boring, but for me, they have been moments of joy. The first person in this research group, I would like to thank is my main supervisor Kerstin Severinson Eklundh. She has kept me on track, and her broad perspective on writing and language tools has made my work much more challenging and stimulating than I ever could imagine when I started my doctoral studies. Viggo Kann has been my second advisor, I am very grateful to him. I really enjoined our collaborative efforts in the development of Granska, especially our negotiations about parsing efficiency and linguistic expressive power. The journey with Granska and its siblings seems to have no end, I am very happy about that. My third advisor, Teresa Cerratto Pargman, entered my research a while after my licentiate thesis. I was a bit surprised of her interest in Granska, but I think I am starting to understand now. I am very grateful for her view on tools, mediation and learning which slowly removed my blinkers.

Rickard Domeij introduced the concept of grammar checking for me, and has been a very good companion since then. Two other persons, Johnny Bigert and Jonas Sjöbergh, have played important roles in my research, and at the right moment. When I have felt that we could not get any further with grammar checking, they have come up with new ideas, and most importantly, they have implemented and evaluated them. I am also thankful to Johan Carlberger for his work with Granska, and his devoted efforts to improve Granska's different modules. Stefan Westlund is the kind of Master's student that you very seldom meet. I am very grateful for his great contributions to Grim's development and maintenance. I am also happy that Petter Karlström came into our project about language tools and second language learning, with new ideas and many important and stimulating theoretical discussions. Hercules Dalianis, Martin Hassel and Magnus Rosell have been very nice colleagues during the years, and Martin has also been a good roommate. I have really enjoined our discussions. Magnus Sahlgren showed me that philosophy of language is very important in language technology, and I am very thankful for our discussions.

I also want to thank the teachers and the students at a university in Sweden, who really changed my view on language tools and their use. Auður Hauksdóttir and the researchers in the Norfa CALL Net have been

very important to me. I am very happy for our meetings. I also wish to thank the team of runners at IPLab, IPLubbarna: Eva-Lotta, Lasse, Staffan, Ulla-Britt, Anders, Helge and Chiara. Maria, Henrik and many other people at IPLab have together with IPLubbarna made my days at IPLab much more joyful. My family with Lena, Axel, Isak, Gert, Ulla and Anders means a lot to me. You have all supported me since I started my doctoral studies.

KTH Nada has been a stimulating research environment, and I have had many interesting discussions with researchers, teachers and students through my work here. My research has been funded by KTH Nada, the Council for Research in the Humanities and Social Sciences (HSFR), the Swedish National Board for Industrial and Technical Development (Nutek), the Swedish Agency for Innovation Systems (Vinnova), the Swedish Research Council (Vetenskapsrådet) and the Swedish Academy (Svenska akademien) through the Swedish Language Council (Svenska språknämnden).

Thank you all!

Ola Knutsson

Stockholm, September 2005

# Contents

# 1 Introduction

Writing and written language play an increasingly important part of many people's lives, on the computer, in word processing environments, e-mail clients and chat rooms. New technologies for text production have emerged the last twenty years, and are used in daily life as well in professional contexts.

Ordinary people including children, teenagers, and grown-ups with different economical, linguistic, social and cultural backgrounds write in the computerized world. They are using the textual form of language more than ever before. New media for communication mediate writing; the size of screens might constrain the length of the written messages. The design of the keyboard, for instance the qwerty-keyboard mediates the errors (Wertsch, 1998). Written communication enhanced by computer networks might shape our writing activities in a more dialogic manner (Severinson Eklundh, 1986).

The role of written communication in education cannot be underestimated (cf. Säljö. 2000). Learning through written language is more important than ever, with the ever-growing mass of textual information. However, the main advantage with written language is the same as when writing systems were first developed; we can receive and send messages without the same constraint on time and space as in spoken language (Moro, 1999).

In a country like Sweden people have to use written language to communicate in many situations. For example in correspondence with the authorities, with teachers in their children's schools, and with the coach in the football club, written language plays an important role. Written language is more or less a pre-requisite for everyday communication, and is necessary if one wants to be part of different communities.

This development of society leads to increased needs for tools to deal with text, both for reading and writing. Using text as a medium to

produce and take meaning in order to transform information to knowledge leaves many people out (Säljö, 2000). This should not be seen as if some persons have the ability and some others do not have the ability to communicate through text in a specific language. Rather, there exists a continuum in which people to a different extent can master textual communication in different genres.

A technology that has a potential to aid users in some of their interactions with written language is language technology. Language technology is a diverse discipline, including many different areas and applications. One potential of this technology lies in the ability to analyze and generate authentic language, although sometimes only limited levels of linguistic processing may be involved.

## 1.1 Introducing language tools

A language tool for writing is here defined as a tool, which automatically gives the users feedback on the language they produce. In current programs this feedback is limited. It relies on information available on the surface of the language the users produce and in pre-programmed language models. The programs are in many cases equipped with a language model that is quite static, and not very adaptive.

In spite of the static view that current language tools provide, they can be used in creative ways if users can learn how to use them. If one compares language tools with other tools like a book, or a dictionary, they open up new ways to interact with language. Even an on-line static dictionary might shape a different view of language than the corresponding printed book. An on-line dictionary with a search tool (a good one) can for instance give the user all words that begin or end with a specific letter or all words that include a specific stem. A hyperlinked dictionary can help the user explore the relations between words in efficient ways.

When one compares a language tool with for instance a calculator, they have some similarities. They are designed as tools to aid the user in something that to some extent can be seen as a larger problem-solving task. However, the calculator is based on algorithms and used for problems that are algorithmic. A language tool is based on algorithms but language use is to great extent not an algorithmic problem, it is a social game governed by certain rules. As a social game, language has many rules that are extremely hard for a computer program to apply; currently only a partial use of low-level rules can be achieved. With this in mind, an important question is: are language tools useful?

## 1.2 How mature are language tools?

Word processors have been facilitated with spelling checkers for a long time now, but still, they are not perfect. They do not find all errors in a text, and they do not always give relevant feedback in form of correct diagnoses and correction proposals. The problem is that even what might look like a minor language problem such as a spelling error is dependent on the communicational context. Grammatical cues from the adjacent words might help in many cases, but some problems need a deeper understanding of the text and the communicative intentions of the user. What might look like a spelling error might be something much more problematic. There is no chance to look into the user's mind, to get an advanced interpretation of an error. The clues for the interpretation of the error rely on textual information, pre-programmed language models and the broad setting that the program is used in.

## 1.3 Research issues in this thesis

An assumption for the research presented in this thesis is that language tools at least to some degree have something to offer both native writers and people learning Swedish through writing. The thesis is based and focused on three research issues: designing, developing and evaluating language tools, and in particular the grammar checker Granska. The three research issues should not be seen as organized in a linear way, rather they are closely intervened and dependent on each other.

### 1.3.1 Designing language tools with the aim to support writing and learning

Language tools that should work in users' writing activities must deal with language use as a process and they must also deal with language that is not error-free. This is even more the case for tools used by learners of the written language. When it comes to second language writers, the errors are more frequent, and the feedback from the program is probably more needed. However, second language learners form a very heterogeneous group of people, which also means that the writing and learning processes within this group are very different.

Furthermore, grammar checking is about errors, and within second language research error analysis has been much discussed. This focus on learners' errors has been very popular since the 1970s, and then for a time very criticized. However, the learners' need for feedback on their errors seems hard to neglect, and a new interest in errors in recent years can be seen in corpus oriented research as well as in research focusing on learning and teaching activities.

### 1.3.2 Developing tools that can work with texts that contain errors

A primary problem when developing tools that should deal with texts that contain errors is how the text should generally be analyzed, i.e. how word classes, phrases and grammatical dependencies should be correctly identified. In order to detect errors, the words in the context of the error must be analyzed in a predictable way, otherwise it is extremely hard to design algorithms that diagnose and correct the errors.

The problem of which sentences that belong to the language and which ones do not can be seen as a very important issue in this thesis. However, the work done has a pragmatic approach, with a focus on tools that should aid the user in writing and learning activities. In such tools another problem arises, the problem of detecting, diagnosing and correcting grammatical errors. Some errors will be unrecognized because the general analysis has failed to analyze the text in such a way that the errors are possible to detect. Other pieces of text that are correct will be recognized as erroneous, these are called false alarms. The issue of missed errors and false alarms are central in the development of language tools.

### 1.3.3 Evaluating language tools in use

There are at least two objectives when evaluating language tools; to get knowledge about their performance in technical terms and to find out if users think that they get help from these language tools in their current writing activities. High performance does not necessary lead to high usability, which means that the more technical evaluations in terms of number of correct error detections must be complemented with user studies. If an error is detected with high precision, but the user does not know how to correct the error anyway, the usability of the error detection algorithm must be questioned. It could be the case that one error only needs to be detected with low precision because of the fact that the error is very important and easy for the user to detect and correct, but other errors need to be detected with a very high performance because their corresponding correct construction is extremely frequent. An example could be an error detection algorithm that tries to distinguish between the correct use of *these* and *those* in English text. Both words are frequent and that will lead to a distressing number of false alarms, even though the performance of the error detection algorithm is very high.

How to study writing and text production when there is a need for a grammar checker is a complicated matter. The writing activity is not easy

to access and to get hold on; it is an intricate web of different socio-cognitive processes including language, genre, and problem-solving constrained by individual, technical, social and cultural factors and norms.

In addition, textual studies of the performance of the language tools developed can give a complementary view of the tools. These studies might be seen as much more straightforward, but they only constitute a different problem space. To analyze, group and annotate errors, and other constructions in text is very difficult, and involves many decisions that cannot be taken from grammar books.

## 1.4 Outline of the thesis

The rest of the thesis is structured as follows.

**Chapter 2: Language tools and writing**

In this chapter the concept of computerized language tools is defined and placed into the context of writing.

**Chapter 3: Designing and developing a grammar checker for Swedish**

There exist several methods and approaches to grammar checking. A starting point for designing a grammar checker is to evaluate current general text analysis tools, and other resources in form of corpora and lexica. In order to detect errors several methods have been developed during recent years, and a spectrum of those are presented in this chapter. In addition, the Swedish grammar checker Granska is presented. Granska is central in this thesis, both as an object of development and as a target of several evaluations.

**Chapter 4: Using language tools in second language writing and learning**

Language tools have been used in second language writing and learning for quite a while. Research in this area leads into a field called computer assisted language learning. This is a broad area including research and knowledge from applied linguistics, second language acquisition/learning, language technology, and artificial intelligence as well as knowledge and experience from language teachers. However, in this chapter I will mainly focus on its relation to language technology and second language learning.

**Chapter 5: Evaluating language tools**
Evaluating language tools involves several methodological considerations. The writers' activities are hard to access, and this raises questions on methods for studying the use of language tools. In this chapter two user studies that have been carried out are presented and discussed. In addition, the textual evaluations of Granska and related technology are presented.

**Chapter 6: Designing a new environment for writing and learning**
One result of the development and evaluations of language tools presented in this thesis is a language environment called Grim. In this chapter Grim is briefly presented.

**Chapter 7: Summary of papers**
The five papers that the thesis is based on are summarized in this chapter.

**Chapter 8: Discussion and conclusions**
In this chapter I return to the research issues pointed out as central in this thesis. Reflections on methodology and ideas about future work are discussed.

## 1.5 List of publications

### 1.5.1 Papers included in this thesis

**Paper 1:**
Carlberger, J., Domeij, R., Kann, V., & Knutsson, O. (2004, submitted). The development and performance of a grammar checker for Swedish: A language engineering perspective.

Paper 1 is partly based on the paper:
Domeij, R., Knutsson, O., Carlberger, C., & Kann, V. (2000). *Granska – an efficient hybrid system for Swedish grammar checking*. In proceedings of the 12th Nordic Conference in Computational Linguistics, Nodalida'99, Trondheim, Norway.

**Paper 2:**
Domeij, R., Knutsson, O., & Severinson Eklundh, K. (2002). *Different ways of evaluating a Swedish grammar checker*. In the proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas, Spain.

**Paper 3:**

Bigert, J., Knutsson, O., & Sjöbergh, J. (2003). *Automatic evaluation of robustness and degradation in tagging and parsing*. In the proceedings of the Recent Advances in Natural Language Processing 2003 (RANLP 2003), Borovets, Bulgaria.

**Paper 4:**

Bigert, J., Kann, V., Knutsson, O., & Sjöbergh, J. (2004). Grammar checking for Swedish second language learners. In P. J. Henrichsen (Ed.), *CALL for the Nordic languages. Tools and methods for computer assisted language learning*. Copenhagen, Denmark: Samfundslitteratur.

**Paper 5:**

Knutsson, O., Cerratto Pargman, T., Severinson Eklundh, K., & Westlund, S. (2005 accepted). Designing and developing a language environment for second language writers. *Computers and Education. An International Journal*. Elsevier.

### 1.5.2 Other relevant publications

Sjöbergh, J., & Knutsson, O. (2005). *Faking errors to avoid making errors: Very weakly supervised learning for error detection in writing*. In the proceedings of Recent Advances in Natural Language Processing 2005 (RANLP 2005), Borovets, Bulgaria.

Bigert, J., Sjöbergh, J., Knutsson, O., & Sahlgren, M. (2005). *Unsupervised Evaluation of Parser Robustness*. In the proceedings of CICLing 2005, Mexico City, Mexico.

Knutsson, O., Cerratto Pargman, T., & Severinson Eklundh, K. (2003). *Transforming grammar checking technology into a learning environment for second language writing*. In the proceedings of the HLT/NAACL 2003 workshop: Building Educational Applications Using NLP, Edmonton, Canada.

Knutsson, O., Bigert, J., & Kann, V. (2003). *A Robust Shallow Parser for Swedish*. Paper presented at 14th Nordic Conference in Computational Linguistics, Nodalida'03, Reykjavik, Iceland.

Bigert, J., & Knutsson, O. (2002). *Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge*. In the proceedings of RObust Methods in Analysis of Natural language Data, ROMAND 2002, Frascati, Italy.

Knutsson, O., Cerratto Pargman, T., & Severinson Eklundh, K. (2002). Computer support for second language learners' free text production – Initial studies. In M. Valcke and A. Bruce (Eds.), *European Journal of Open and Distance Learning (EURODL).* Also in the proceedings of ICL2002, 5th International Workshop on Interactive Computer Aided Learning, Villach, Austria.

Knutsson, O. (2002). *Inkongruens i predikativ – både rätt och fel.* In the proceedings of Svenskans beskrivning 25, Åbo, Finland.

Knutsson, O. (2001). Automatisk språkgranskning av svensk text. Licentiate thesis, TRITA-NA-01-5, ISBN 91-7283-052-2, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Domeij, R., Knutsson, O., & Öhrman, L. (1999) *Inkongruens och felaktigt särskrivna sammansättningar – en beskrivning av två feltyper och möjligheten att detektera felen automatiskt.* In the proceedings of Svenskans beskrivning 24, Linköping, Sweden.

## 1.6 Project framework

Development of large-scale language technology systems is to my knowledge always based on teamwork. A system that is presented in this thesis is the grammar checker Granska[1], and other applications developed from this system. In addition, several evaluations of Granska have been carried out. Granska has evolved through the years and a lot of people have been working with the system. In a first phase during the years 1996-1997 a first prototype, a writing environment called Granska was developed by a project group led by Kerstin Severinson Eklundh. Rickard Domeij and Stefan Larsson worked in the project. I participated in this project through my Master's thesis and as a part-time worker in the project.

In 1998 a new project was formed called "Integrated Language Tools for Writing and Document Handling" also led by Kerstin Severinson Eklundh. This project was a collaboration project between KTH Nada and a group at the Department of Linguistics at Göteborg University and a group at the Swedish Language Council. Vinnova funded this project. In 1999 I started my doctoral studies within this project.

[1] The word Granska means in Swedish to scrutinize.

*Project framework*

---

At KTH Nada the work focused on the design and improvement of writing environments through user studies and a new grammar-checking engine also called Granska. In the following when I refer to Granska it is this version of Granska I mean. Viggo Kann led the smaller research group focusing on the development of the grammar-checking engine. The group at KTH Nada consisted of me, Johan Carlberger, Rickard Domeij and Annika Hansén-Eriksson. Apart from the software developed this work also resulted in my licentiate thesis (Knutsson, 2001). The work with Granska is presented in Paper 1. Improvements and evaluations of Granska made after this project have been added to the same paper. A user study and a textual study carried out by me in this project are presented in Paper 2.

In 2001, a project called CrossCheck with a special focus on grammar checking for second language writers of Swedish was started, funded by Vinnova. This project, headed by Viggo Kann, was a collaboration project between KTH Nada and Department of Linguistics at Stockholm University. The project consisted of two parts; the first focused on methods for developing and evaluating tools that should analyze and also detect errors in second language writers' texts. The second part of the project focused on learner corpora and the collection of such material. Project members at KTH Nada were Kerstin Severinson Eklundh, Teresa Cerratto Pargman, myself, Johnny Bigert and Jonas Sjöbergh.

Important improvements were made on Granska, and a server version of the program was also an important contribution, which made it possible to develop different user interfaces to the same grammar checking engine. The development of two new methods and programs for grammar checking called ProbGranska and SnålGranska were also carried out in this project. In connection to the development of ProbGranska a partial and shallow parser was developed, called Granska Text Analyzer (GTA). The evaluation of GTA is presented in Paper 3. The two new methods for grammar checking are presented in Paper 4.

In the corpus part of the project, a corpus called the CrossCheck corpus was collected, compiled and made available for research purposes. A subset of the CrossCheck corpus was collected by me in a sister project called "The use of language tools for writers in the context of learning Swedish as a second language". This subset was used in a comparative evaluation of Granska and the Swedish grammar checker in Microsoft Word carried out by me and presented in Paper 1.

Teresa Cerratto Pargman leads the project mentioned above which is funded by the Swedish Research Council (Vetenskapsrådet). Other project members are myself, Kerstin Severinson Eklundh, Petter Karlström and Stefan Westlund. This project focuses on the use of computer language tools for writers in the context of learning Swedish as a second language. We study how learners develop their writing practices in the context of learning Swedish as a second language, how they use available writing tools and how these tools shape learners' understanding of the new language. An important contribution of this project is the language environment Grim. Grim is a program based on design principles identified through field studies with Granska and second language learners participating in an advanced course at a Swedish university. Grim is also to a great extent based on the results and the different language tools developed in the projects described above. The field study and the design of Grim are presented in Paper 5.

Through the projects I have supervised bachelor and master students, and their work have of course contributed a lot to my work. Lena Öhrman studied split compounds, and the use of Granska on the second language learners' texts. Anna Staerner made a study on grammar checkers in learning environments with special focus on how to detect word order errors within the Granska framework. Jens Eeg-Olofsson studied error annotation and the use of prepositions in second language learners' texts and grammar checking. Magnus Johansson took the first steps on machine learning and error detection, and built annotation and evaluation tools. Victoria Johansson evaluated the phrase detection capacity of the first embryo of the partial and shallow parser GTA. Henrik Hahne evaluated the clause boundary detector in GTA. Ola Karlsson studied the use of Granska by second language writers at work places. Anna Tyndall investigated how language technology could be used in editing tools. Ylva Stenervall developed a more interactive web-based interface to Granska. Stefan Westlund developed the web-based language environment Grim as his Master's project. However, Westlund's work continued on voluntary basis and resulted in a robust application running on most operating systems, and with many users all around the world.

### 1.6.1 Contributions and collaborations in the included papers

**Paper 1**

Main contribution of the paper: The design and development of the grammar checker Granska and its evaluations. A limited number of grammar checking rules covers quite a few error types. The language

engineering perspective is well represented and is shown in a fast grammar checker, which can handle a lot of error detection rules. The integration of the spelling checker Stava (Kann *et al.*, 2001) in the grammar checker was a good design decision according to the evaluations made.

The design of the Granska grammar checker engine is based on teamwork, with competence from computational linguistics represented by myself and Rickard Domeij and competence from computer science by Viggo Kann and Johan Carlberger. I designed the rule language. The implementation of the Granska system including the part-of-speech tagger, rule language, and word form generator were work carried out by Viggo Kann and Johan Carlberger. I wrote the linguistic rules in Granska. I also made the two evaluations in this paper. I, Rickard Domeij and Viggo Kann wrote the paper.

**Paper 2**

Main contribution of the paper: The main idea is to show how different methods for evaluating the Swedish grammar checker Granska broadens the understanding of its performance. An important argument put forward is that both user studies and textual evaluations are needed when evaluating language technology.

I carried out the first two studies presented in this paper. Rickard Domeij carried out the third study. Kerstin Severinson Eklundh supervised all three studies. Viggo Kann supervised me in the textual evaluation. I, Rickard Domeij, and Kerstin Severinson Eklundh wrote the paper.

**Paper 3**

Main contribution of the paper: An automatic method for evaluation of parsers' and taggers' robustness against ill-formed text. The experiments presented showed a graceful degradation in performance for both taggers and the partial and shallow parser GTA.

I carried out the corpus annotation for the parser evaluation. Johnny Bigert and I developed the parser GTA that is presented and evaluated in this paper. I wrote the linguistic rules and Johnny Bigert programmed the disambiguation algorithm, and the interface to Granska as well as the XML-generator for the parser output. Jonas Sjöbergh ported the different taggers to Swedish and connected them to the GTA parser in the Granska framework. Johnny Bigert and Jonas Sjöbergh carried out the practical work with all different data sets used in the evaluations. Johnny Bigert and I wrote the paper.

**Paper 4**

Main contribution of this paper: Three different methods for grammar checking are presented and compared: Granska, ProbGranska and SnålGranska. An ensemble of these grammar checkers will increase both recall and precision. When all three grammar checkers agree on error detection, precision will be extremely high. Granska is based on mainstream language technology, but with special focus on language engineering. ProbGranska and SnålGranska are methods which are more novel, but not totally unique. In the evaluation, the Swedish grammar checker in Microsoft Word was used for comparison.

The development of Granska has been presented in the description of Paper 1, but Granska has been improved by all four writers of this paper. Johnny Bigert developed the main algorithms in ProbGranska. I developed the transformation rules used in ProbGranska, which are important in order to get rid of false alarms. I and Jonas Sjöbergh developed the original idea of SnålGranska, and Jonas Sjöbergh implemented it. Jonas Sjöbergh also carried out the comparative evaluation of the three grammar checking methods. Johnny Bigert, Viggo Kann, myself and Jonas Sjöbergh wrote the paper.

**Paper 5**

Main contribution of this paper:
The study of the use of Granska in second language writers' free text production, and its role in their writing and revisions are important contributions of this paper. A new method for these kinds of studies was developed, which focused on the users' judgments and comments of the feedback from the language tool. In addition, several design principles for second language learning environments based on language technology were developed, and which were realized in Grim.

The field study was designed and carried out by me and Teresa Cerratto Pargman. The method for studying writers' free text production when using Granska was developed by me, and supervised by Teresa Cerratto Pargman and Kerstin Severinson Eklundh. I carried out the analyses of the data collected. The design of Grim was teamwork, but the influence from its programmer Stefan Westlund is very strong. I, Teresa Cerratto Pargman and Kerstin Severinson Eklundh wrote the paper.

# 2 Language tools and writing

Written language has a long history, and is far from being transcribed speech (Olson, 1995). Written language is instead a model of spoken language; it brings many linguistic concepts, like words, into awareness (Lantolf, 2000). The model is provided by the writing system and the conventions of written text (Olson, 1994; Wertsch, 1998). This spin-off effect of written words has according to Lantolf (2000) many similarities with models of cognition that would not have existed without the advent of the computer. The point is that writing and computer systems were not designed as models for language and cognition; they have become models, and they can be good or bad models. Lantolf gives examples of many linguistic features that are not represented in written form, for instance intonation and stress. This is important when building tools for written language, and Källgren (1979) has put it very well: "what makes a text complicated, is not only what actually is written, it is also what is not written" (my translation). Language tools for writing do not deal directly with the user's language, but with its written form.

Learning to write in a first or a second language should thus be seen in the light of learning the model of the specific language through its writing system. From this point of view we might also see the potential of writing when learning a language – as a tool that turns language into objects of reflection and analysis (Olson, 1995). Studies of novice writers have shown that cognitive work has spilled into the text (DiCamilla & Lantolf, 1994). Different kinds of grammatical forms are written in the text not for communicative purposes but for the writer's own control of thoughts and the text, so called private writing. This means that writing and language are not only used for communication, they are also used as cognitive tools.

Tools for written language have been developed through the years, and have been competing but also replaced, where the computer's replacement of pen and paper must be seen as revolutionary. This process cannot be seen through a simple replacement model (Haas, 1999). The pen and the computer still co-exist, but are used for different purposes.

The tools that are developed are filled with cultural and social features/structures as well as individual practices. Haas embraces Vygotsky's focus on the mediational role of tools and puts it:

"Because literacy technologies are mediational, they stand in a central position within the conduct of literacy, mediating in a sense between the individual actor and the cultural and historical milieu within which that actor works" (Haas. 1999. p. 212).

According to Wertsch (1991), Vygotsky saw the limit of the analogy between psychological tools (e.g. language) and technical tools (e.g. tools to write with), but Wertsch (1991) argues in the other direction, by saying that Vygotsky did not push the analogy far enough. Wertsch (1991) focuses on the diversity of tools, and that the mediational means should not be seen a single whole, but as a tool kit. Thus, language should not be viewed as a homogenous essence, and language development should not be seen as single unified process. This is important to consider when we try to understand how language tools interact with real language use and learner's language development. To be more concrete, current language tools are more or less static, and cannot adapt to different text genres, or different levels and ways of language development.

## 2.1 Defining language tools in the context of writing

Language tools designed as writing aids are in fact a quite diverse set of tools. They can be viewed along several dimensions, but the distinction between active tools and passive tools might be one of the most important. Active tools are tools that do something with the language and this includes simple tools like for instance word counting and hyphenation in word processors. In order to count words or hyphenate them a computer program needs to transform the sequence of characters into words. This is a first step in the automatic processing of written language, which must also be seen as an active one. The program carries out a linguistic task for its user.

Spelling checkers are more advanced, although most of them include only a model of word composition in isolation and no language model. More advanced language tools like grammar checkers, text summarizers and machine translators may give an impression of having very active language properties, but their "understanding" of language is still very limited.

Passive tools like a digital grammar book or an on-line dictionary are more or less used as ordinary books, but the computer can change the

way users can interact with them. Digital corpora can also be used as tools if they are used with efficient search programs and if they are integrated with the linguistic task at hand.

If these tools are integrated with linguistic material on the user's computer, they can take a more active role. Users can interact with language through language tools, not necessarily mediated by language but through a "dialogue" involving a meta-language, examples of language use, graphical representations or through other users. The distinction between active and passive is thus not only a question of the language tools' ability to "understand" language; it is also a design issue concerning interaction and dialogue.

## 2.2 What kind of structures are language tools based on?

If we assume that language tools have a mediating role in users' writing, we must also try to uncover the structures and knowledge "embodied" in these tools. This is a part of what Haas calls "The historical-genetic method" based on Vygotsky's ideas (Haas, 1996, 1999), which includes the historical study of how the tools were developed. The other parts of this method include how the tools are transformed by use, and the transformative power of tools on consciousness.

Language tools do not differ from many other tools in that they are products of science, and social, cultural and technological development. And as with many other tools the developers of the tool have a specific user and activity in mind when developing the tool. In our case it is writing and the writing processes that have been more or less modeled. This concerns the user interface, and more deeply the interaction between the user and the language tool, and the kind of language the developers of the tools assume that writers use.

In order to make the language tool more concrete we will in the following focus on spelling and grammar checking. In this context several questions arise concerning a tool's design, its surface, its possibility to interact, and the language engine inside the tool. MS Word's grammar checker can be used as an example. When this tool is analyzed, questions about its design can be asked; for instance if the design is based on research on human computer interaction and writing processes or if the design is implemented because it is now possible through more efficient algorithms and faster personal computers?

In the Swedish versions of the Microsoft Word's grammar checker errors violating several features are treated in a step-like fashion, feature by feature. An agreement error like: "Jag såg **ett lilla hund"** (Eng. I saw a little dog) is proposed to be changed to "Jag såg en **lilla hund"** which therafter is proposed to be changed to the correct sentence "Jag såg en liten hund".

The question arises whether this way of designing the user's interaction with the grammar checker is a product of knowledge about writing processes, or is it just a design constrained by limitations of the current technology?

The developers' view of language tools is constrained by factors like the specific kind of language technology used, and what it is based on. The developers' own beliefs of language use also play a part. In the Swedish case, a very well recognized word list of what is considered as the main vocabulary of Swedish is the Swedish Academy's wordlist (SAOL, 1986). This wordlist mediates to a great extent the correct phonological mapping of words in Swedish. If an item does not exist in this list it might create doubt in a person using the wordlist for checking spelling.

When it comes to language tools, in this case a spelling checker, several new layers (or functions) mediate the act of spelling checking. The spelling checker has to fulfill the tasks normally carried out by humans when they use a wordlist for checking or are required to. In this Swedish case, it concerns for instance to add inflections to words in the wordlist (which normally associates two inflections with the lemma) and "rules" for the building of compounds.

Although we want to design language tools just as tools, and not as tutors (Levy, 1997) there will exist some kind of user modeling. We must assume that the writers want to get help with a specific activity and use more or less an "ideal" form of language. In addition, we must assume that the language tools should interact with the users' writing processes in limited ways. This is user modeling, but it is hopefully a weak version, not including any techniques for user adoption of for instance error catalogue, genre, and personal preferences about explicit or implicit feedback. To conclude, designing and developing language tools for writing is based on several idealized conceptualization of writers' activities, language use and preferable ways to interact with a computer. In order to understand the role of language tools in the context of writing, a long-term goal must be to try to uncover these idealizations.

# 3 Designing and developing a grammar checker for Swedish

Research on grammar checking of English has a quite long history, and how to handle ill-formed input in general NLP-systems has also been studied for a long time (Jensen et al., 1983). Current research for English seems to have two different areas: machine learning (Chodorow & Leacock, 2000; Han *et al.*, 2004; Mangu & Brill, 1997) and grammar checking for second language learners of English (Han et al., 2004; Park *et al.*, 1997; Schneider & McCoy, 1998; Tschichold *et al.*, 1997). For other languages mostly native speakers have been in focus. Several languages have been targeted like Dutch (Vosse, 1994), Spanish and Greek (Bustamente & León, 1996), Swedish (Arppe, 2000; Birn, 2000; Carlberger *et al.*, submitted; Domeij *et al.*, 2000; Sofkova Hashemi, 2003; Sågvall Hein, 1998), Norwegian (Bondi Johannessen *et al.*, 2002; de Smedt & Rosén, 2000), Danish (Paggio, 2000) and German (Bredenkampf et al., 2000) just to give some examples of European work in the field.

Some research and software development within the field of grammar checking has also been focused on so called controlled languages. A controlled language is a subset of a human language with constructed constraints on syntax and lexicon (Almqvist & Sågvall Hein, 1996).

Methods in grammar checking are also useful in many other applications in language technology, for example machine translation, text summarization, text generation and many other applications from OCR error handling to text-to-speech systems (Han et al., 2004).

## 3.1 Designing a grammar checker

Designing a grammar checker involves many issues, but the user's needs must be seen as the central issue. This includes not only the user's language, but also how the user is going to use the grammar checker, for what purpose and in what kind of activity. If the context is professional writing, the grammar checker might be designed in one way; if the

grammar checker is going to be used in learning a second language it might be designed differently, i.e. learners and professionals have different goals when they write. The way the user's errors are analyzed is important when designing what kinds of feedback that will be given.

However, technology imposes a lot of constraints on what kind of grammar checker it is possible to develop. When designing a grammar checker, current technology that the grammar checker should build on must be evaluated. In order to detect errors the underlying linguistic analysis must be robust against errors, which means that ungrammatical sentences must be given some analysis and not be left without any interpretation. For many languages, the linguistic resources, e.g. corpora of correct language and error corpora are very limited. Tools for analyzing text might also be underdeveloped. Therefore, cost-effective development methods have to be chosen when developing a grammar checker for other languages than English and a few more. The technology chosen, or the one to be developed, must also be efficient if the grammar checker should be used in interaction with users during writing. In addition, the possibility of tuning the grammar checker's level of performance is an important requirement if it should be possible to use the grammar checker for different writing tasks. A user might want to find as many errors as possible with the cost of many false alarms in one text, while in other texts it is only important to get rid of the most annoying errors.

One central problem that arises during the design and development of a grammar checker is how to judge grammaticality. The grammaticality judgment has to be done by both the user and by the program and on several levels. A user can be good at judging the grammaticality and then false alarms and limited feedback might be of limited concern, but if the user has problems in judging the grammaticality then false alarms and limited feedback might be a greater problem. The ability to judge grammaticality is what determines the program's performance on detecting errors.

### 3.1.1 Grammar checking and grammaticality

The question of grammaticality and acceptability is a research area of its own, and has concerned linguistics for several decades. The famous sentence by Chomsky (1957) "Colorless green ideas sleep furiously" is a good example of a sentence that a grammar checker must pass through. The sentence is grammatical but it is not acceptable.

A piece of language can be acceptable in one context, but unacceptable in another context. The meaning of this piece of language has to be taken into account to make the correct judgment. Manning proposes that we should look at *form*, *meaning* and *context* (Manning, 2002) when trying to model grammaticality. Manning seems to avoid the notion acceptability, finding it hard to define. His view focuses on language use and evidence from text corpora, and he does not seem to separate syntax from meaning and context.

### 3.1.2 A problematic example: Split compounds

Split compounds are frequent in Swedish texts, and they also demonstrate the critical relations between form, meaning and context. If a split compound creates an ungrammatical structure it must be an error; but if the construction is grammatical but for humans not acceptable, some people think that it still should be signaled as an error. We can see that the interaction between form, meaning and context is important when judging grammaticality and acceptability. For current grammar checkers the division between grammaticality and acceptability is difficult to deal with. Four possible categories of constructions arise based on the binary distinctions *grammatical* and *acceptable* (James, 1998):

**+grammatical +acceptable**: *Hon åt en grön sak* with the meaning *She ate a green thing*. This sentence cannot be ungrammatical, but it can be unacceptable, see next example.

**+grammatical -acceptable**: *Hon åt en grön sak* with the intended meaning *She ate a vegetable* (Swe. grönsak). If the writer means or if the context indicates a vegetable and not a green thing the sentence is unacceptable.

**-grammatical -acceptable**: *Hon åt inte grön saker* (with the intended meaning *She did not eat vegetables* (Swe. grönsaker). This sentence is both ungrammatical and unacceptable, and it has two different interpretations, either as a split compound or as an agreement error within the NP “grön saker” (green things).

**-grammatical +acceptable**: It is hard to find a split compound that is ungrammatical but acceptable – grammaticality is a prerequisite for acceptability.

### 3.1.3 How to view the errors and to provide feedback

Within the field of error analysis there seems to exist some consensus about the fact that errors can be viewed in at least two ways; they can be diagnosed and they can be described (James, 1998). When an error is diagnosed its human causes are taken into account. The error description focuses on formal aspects, what kind of structural violations the errors causes. In grammar checking these two ways of viewing errors and giving feedback to the user coexist. A so-called context-sensitive spelling error refers to the user's spelling ability, its causes. An agreement error is a good example of an error description. This mix of feedback might be unfortunate, but my opinion is that it is a result of developers' and researchers' strive to present the programs' functions in a comprehensive way.

A context sensitive spelling error like "Det var en vacker frö" (Eng. That was a beautiful seed) with the intended meaning "Det var en vacker för" (Eng. That was a beautiful prow) can in Swedish be interpreted as an agreement error. By using the diagnose *context sensitive spelling error* the assumption is that writer has made a spelling mistake, and the only way to understand that is to examine the context. By using the feedback based on the error diagnosis the grammar checker is making a much stronger claim about the error than it would do using the error description. Henceforth I will use the term error diagnosis in order to simplify.

Less problematic than diagnosis and description is error detection, which means to signal that a sentence or segment is erroneous. More problematic is error location. Some errors are easier to locate for instance disagreement within a noun phrase. More difficult to locate are constituents that are missing in the sentence, like a missing verb or subject. In grammar checkers the two concepts are used together, but in the following I will use only the term error detection.

The last step in this feedback process on errors is error correction. A teacher might give error correction, but I think that providing correction proposals is what a grammar checker should do.

### 3.1.4 Different ways of providing feedback on errors

There exist at least three different ways to provide the user with feedback in current grammar checkers:

- Batch mode. The user gets an error report on all errors found in the text, i.e. an output file with all errors in the text described, see Figure 1.
- Sequential mode. The user gets reports on errors from a certain point in the text, one error report at a time, sentence by sentence. A user that uses this mode of interaction will get feedback on all errors from point A to point B, see Figure 2.
- Continuous checking mode. The program marks all errors that it encounters. The user can choose between the different markings in the whole text in order to get diagnoses and correction proposals using the mouse, see Figure 3.


Figure 1. *Getting feedback on errors in batch mode. This is the output from a web interface to Granska called WebbGranska.*

For different activities, different modes of interaction might be suitable. A journalist might want an error report resembling the one from a proofreader. A second language writer might want immediate feedback on the errors using the continuous checking mode, because using the correct forms is important when formulating the rest of the text. Another user might want to be sure that she has judged all alarms from the

grammar checker, and scrutinizing them in a sequential mode might be the best way to do that.

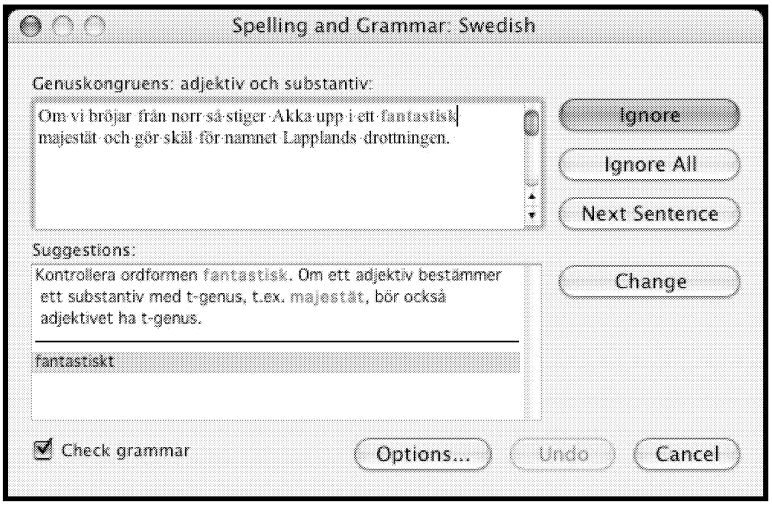


Figure 2. *Getting feedback on errors in a sequential way. Dialog box from the Microsoft Word's grammar checker.*

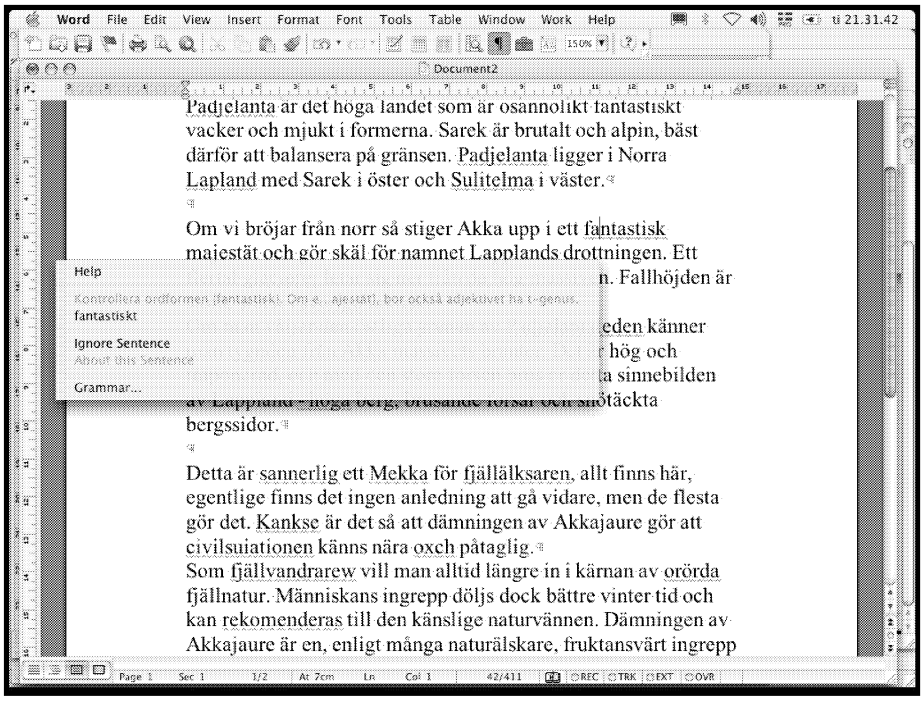


Figure 3. *Getting feedback in continuous checking mode in Microsoft Word. Errors are underlined with red and green. Diagnoses and correction proposals are invoked by using the mouse and pop-up menus.*

### 3.1.5 Grammaticality judgments made by the user

The user has to judge the grammaticality of the language fragment that program gives an alarm on in order to decide what to do with the current alarm. The ability of the user to judge grammaticality is important for the user's benefit of using the grammar checker. If the user has problems in the judgment process, many alarms from the program demand heavy work with revision (Domeij 2003; Paper 5). The ability to judge grammaticality seems to be much stronger in first language learning than in second language learning (Ellis, 1997). In the following focus will be on how grammaticality judgments are made by the different modules in a grammar checker and how they are communicated to the user. In order to discuss this matter the basic functionality in a grammar checker first has to be described.

## 3.2 Basic functionality in a grammar checker

The main functions of a grammar checker are to detect, diagnose, and propose corrections on errors (Domeij 2003; Paper 5). In order to detect, diagnose and propose corrections on errors in a text some basic functionality is needed. The first linguistic analysis that has to be carried

out is tokenization, which means to divide the text, a sequence of characters, into sentences and words. This process might seem trivial, but it is actually not a trivial task (Grefenstette & Tapanainen, 1994) and is a source of errors when processing text. Examples of problems when tokenizing Swedish are the interpretation of abbreviations and hyphenation. Current programs for text analysis include a so-called tokenizer, and most of them are rule-based (Grefenstette & Tapanainen, 1994). Some systems contain an idiom matcher (Birn, 1998) in order to analyze multi-word expressions as one token, e.g. "i dag" (Eng. today).

The next processing step in many systems is some kind of morphological analysis. One exception to this is an approach based on Latent Semantic Analysis (Jones & Martin, 1997), in which the only "general" linguistic analysis is tokenization. The morphological analyzer assigns possible word classes to the words together with corresponding morphological information e.g. number, gender and species for Swedish nouns. This analysis is in most cases based on a lexicon together with rules such as in two-level morphology (Koskenniemi, 1983) or by a lexicon and statistics like in Carlberger and Kann (1999). For unknown words not in the lexicon, heuristics or statistics is used to determine one or several word classes of the unknown word (Carlberger & Kann, 1999).

From this point in the pipeline of linguistic processing some systems have a separate module for disambiguation of word classes, i.e. deciding which word class a word belongs to in the current context. A so-called part-of-speech tagger carries out this disambiguation using statistics (Brants, 2000) or by using rules (Karlsson, 1990). In other systems all syntactic analysis is taken care of in the same module, which is called a syntactical parser. When it comes to the syntactic analysis carried out by a syntactical parser there exist two main perspectives: systems identifying the phrase structure of the sentence and systems identifying the dependency structure of the sentence. Both methods have been used in grammar checking, phrase structure in for example the English grammar checker Critique (Ravin, 1993; Richardson & Braden-Harder, 1993) and dependency structure in another grammar checker for English described by Bourdon (1998).

A module for some kind of semantic analysis is very rare in grammar checking systems, and has only been used in a very limited way under certain conditions (Heidorn, 2000). Statistical lexical semantics has been tried in very limited domains of error detection (Jones & Martin, 1997). More advanced linguistic processing has to my knowledge not been

attempted in the field of grammar checking. To conclude, most approaches are purely syntactic, and quite partial and shallow.

### 3.2.1 Methods for error detection

So far only methods for general language analysis and their application in grammar checkers have been discussed. However, almost all systems need some modules that detect errors in the sense of pointing out the errors' positions in the text. Methods for error detection can as well as many other methods for language technology be divided into three groups: rule-based, statistical and hybrid.

The first group contains rule-based methods, which are the traditional way to deal with grammatical problems. The most ambitious approach is to try to test if a sentence is generated by a grammar or not. This is a full and deep parsing approach based on hand-crafted rules and could have been the ultimate solution to grammar checking. If a system could divide sentences into grammatical and ungrammatical ones very much would have been gained. However, current methods do not seem to have the ability to deal with this problem. In a system like Critique (Richardson & Braden-Harder, 1993) the deep analysis which includes semantic analysis is left out, and replaced with heuristics. When the full analysis fails, e.g. when it is not possible to build a complete syntax tree, a technique called relaxation is used. Relaxation means to loosen up some constraints in order to let for instance an agreement error through. If this also fails so called parse fitting is used, which means to build a syntactic structure from the longest constituent (Jensen et al., 1983). Relaxation can be seen as a phenomena-based approach, because the grammarian needs to program which constraint should be relaxed. If the constraint on agreement with the noun phrase is relaxed, the error (the phenomenon) must be an agreement error within the noun phrase. To get hold on word order errors the fitted parse must be examined with special rules, which must also be sorted in under the paradigm of phenomena-based grammar checking. The Swedish grammar checker Scarrie (Sågvall Hein, 1998, 1999) is based on a relaxation technique and local error rules. This system parses as much as is needed and then searches the output from the parser Uppsala Chart Parser (Sågvall Hein, 1981) for errors.

Phenomena-based approaches to grammar checking are also the most frequently used ones. Systems with the aim to give a diagnosis or a description must in some way focus on specific phenomena in order to give the user a diagnosis and a correction proposal. Many of these systems do not try to parse the whole sentence, instead they search for the error with help of the information from part-of-speech tagging and partial

parsing. An attractive exception, although belonging to this group, is a technique called positive error detection, where the difference between one broad positive rule and the union of two narrow positive rules forms the search expression for an error (Sofkova Hashemi *et al.*, 2003).

The second main group contains statistical methods where statistics plays an important role in the decision of grammaticality. One of the first experiments with statistical methods to detect errors in text was proposed by Atwell (1987). He proposed the use of the probabilities calculated within a part-of-speech tagger. Part-of-speech transitions with a very low probability were considered as errors. The results of his experiments are hard to judge because of a very limited evaluation. A different approach was proposed by Bigert et al (Paper 4) and Bigert & Knutsson (2002), in which the basic error detection algorithm uses part-of-speech tag trigram frequency information gathered from a corpus of proofread language. When checking a new text, unseen trigrams according to the corpus are considered as erroneous.

Statistical methods can also be characterized as supervised or unsupervised methods. This distinction is made to tell if the errors are explicitly pointed out for the machine learning algorithm. This is denoted supervised learning, in contradiction to unsupervised where no information about the errors is given to the system. Using annotated errors from a learner corpus is supervised learning (Izumi et al., 2003). Using corpora without annotations of errors and methods for comparing different data sets must be considered as unsupervised (Bigert & Knutsson, 2002; Chodorow & Leacock, 2000). Between these two poles, work has been carried out where the errors are created by rules, and thereby automatically annotated, and this approach can be seen as weakly supervised (Sjöbergh & Knutsson, 2005).

Within the field of machine learning there is an extreme case of phenomena-based grammar checking, with a focus on the problem of choosing the correct word in a set of words that are commonly confused, so called confusion sets (e.g. Swedish: dem, dom, de, English: to, too, two). This is commonly called context-sensitive spelling error detection and correction.

Most research has dealt with around 20 confusion sets (in total approximately 40-50 words), and reports an accuracy of around 90%. However, the targeted forms might be frequent, and important in converging to the written norm of a genre. When a new confusion set is added to “the checker” this might sink the whole approach. I have not

found any research trying to answer if such a limited set of error forms is useful. Carlson et al (2001) show how the method could be scaled up, and they also point out that accuracy must increase up to 98-99% in order to be useful, because of the fact that the words in the confusion sets are very frequent, and frequent false alarms on these words will be annoying for the users.

There are at least two scaling problems in this paradigm, automatic scaling of the number of confusion sets (Huang & Powers, 2001), and scaling the learning methods to handle a large amount of confusion sets (Carlson et al, 2001).

I have not found any work on Swedish and disambiguation of confusion sets. However, for Danish some research has been carried out (Hardt, 2001)

The third group is hybrid methods that combine several methods to make decisions on grammaticality and error detection, such as the program ProbGranska. This program combines statistical methods with partial parsing and transformation rules to achieve better performance in error detection, and is presented in Paper 4.

Hybrid methods could also be seen on a system level, where different modules in the grammar checker are based on different frameworks. One example of this is the system Granska, using a statistical method for part-of-speech tagging and a rule-based method for error detection (Paper 1).

A grammar-checking program can be based on several methods, both in making general grammatical judgments and in pointing out the specific error. In addition, modules that diagnose and provide corrections of the errors are also necessary. In phenomena-based grammar checking the detection, diagnosis and correction are very closely related, in many cases parts of the same rule. In a grammar checker such as SnålGranska (Paper 4), where grammar checking is seen as a tagging problem, an error tagger for each error type is built. A tagger designed for only one error type can certainly diagnose errors belonging to that error type. In order to provide the user with correction proposals, special error modules have to be built. They can be quite easy for the error type split compounds by combining the word parts to one word, and more complicated for e.g. agreement errors, where the generation of the correct forms have to be made.

Most research has focused on methods for error detection and diagnosis, and to a limited extent on error correction. This does not mean that error

correction is trivial, instead it is in many times quite difficult. The correction proposal must be automatically generated, and the proposal should make the sentence grammatical. In addition, it is possible to correct an error in several ways. One example is the agreement error "en bilar" (Eng. a cars), which for instance can be corrected to "en bil" (Eng. a car) and "några bilar" (Eng. some cars).

### 3.2.2 Grammaticality judgments made by the general language analyzers

The program's components for general language analysis have to judge the grammaticality of each sentence when they disambiguate between a possible grammatical interpretation and an ungrammatical interpretation. Without a semantic component grammatical but unacceptable constructions will pass the process of grammar checking. Regardless of whether the language model is rule-based or statistical, grammatical but unacceptable constructions will in many cases be ranked higher than ungrammatical constructions. The Swedish sentence "Jag såg det man" (Eng. I saw the man) might work as a simple example. The error cannot be detected partly because of the fact that "det" (Eng. the/it) is quite common as a pronoun in the SUC corpus (Ejerhed *et al.*, 1992) 16311 times as pronoun, 4420 times as determiner. The word "man" (Eng. man/one) on the other hand is quite common as a pronoun (4599 times), and 2600 times as a noun. A plausible tagging of "det" and "man" is therefore as pronouns, such as in the sentence "Jag såg det man behövde se" (Eng. I saw what one needed to see). Language models are nearly always based on correct language. Ungrammatical sentences are neither the base for the grammarian nor existent in the corpus used for the statistical processing. This means that many words that are ambiguous with respect to part-of-speech, will get a higher ranking as grammatical compared with an ungrammatical interpretation.

This disambiguation process sets the limit of which errors a grammar checker can detect. An erroneous phrase can get an ungrammatical interpretation in one context, but a grammatical one in another context. This can make the user confused, especially if the user tries to learn the "behavior" of the program.

### 3.2.3 Grammaticality judgments made by the error detection component

If we assume that the general language analyzers in the grammar checker have let the ungrammatical interpretation go through to the error detection component, then the error can be detected, diagnosed and

corrected, otherwise it is very hard to identify the error. However, all errors will not be detected even though they are in the set of targeted errors. New constraints on the grammaticality judgments are found in the error detector. An error could be very similar to a grammatical construction, e.g. “ett gatans parlament” (Eng. a parliament of the street) vs. “ett hunds mat” (Eng. a dog’s food), which both at the local level could be viewed as agreement errors, but only the latter example is erroneous.

To write a rule that finds e.g. a split compound in Swedish text is not very difficult, it is basically to search for a noun followed by another noun in the text, which has been generally analyzed. This could for example be the nouns in “Jag stod på en **bergs topp**” (Eng. I was standing on a mountain summit) which forms a true split compound, an error. However, such rule will generate a lot of false alarms, e.g. the nouns in “Jag stod vid ett **bergs skuggsida**” (Eng. I was standing on the shadow side of a mountain) will be detected with the same rule. The difficult part is to formulate constraints on the context of the general error pattern. Current error detection components can thus make use of form and local context. However, analysis of words’ or phrases’ current meaning should be very useful in a grammar checker, but this has been rarely used in grammar checkers as discussed above.

When it comes to the generation of correction proposals, new judgments of grammaticality should preferably be made on the sentence with the results of the correction. One way to do this is to check the new sentence with the grammar detection module as made in Granska (Paper 1). Other tools like a full parser or a statistical measurement of grammaticality could be used in a similar way (Bigert, 2005).

## 3.3 Developing a grammar checker for Swedish

It is currently hard to see that a method for grammar checking will be developed that does not have to rely on some kind of grammatical analysis. Statistical methods and a billion-word corpus might work to some extent, but only future research can reveal this. However, current methods rely heavily on grammatical analysis, even though it is not very deep. All approaches have at least in common an analysis of word classes and inflectional information. The only exception is research on disambiguation of so-called confusion sets, where at least some experiments have been carried out without a part-of-speech tagger (Jones & Martin, 1997).

Early research on syntactical analysis of Swedish focused on knowledge poor methods (Brodda, 1983; Källgren, 1992). The Uppsala Chart Parser (UCP) (Sågvall Hein, 1981) accomplishes a more full syntactic analysis. UCP has been used in several applications, for instance in machine translation (Sågvall Hein *et al.*, 2002). Experiments with parsing and so-called field grammars has also been carried out (Ahrenberg, 1990). A deep parsing approach came from Gambäck's work on the Swedish version of the Core Language Engine (Gambäck, 1997). This knowledge rich approach focused on a full and deep syntactical analysis within limited domains.

With the advent of the manually tagged corpus called Stockholm Umeå Corpus (SUC) (Ejerhed et al., 1992) there was an increased interest in methods exploiting this resource, and in particular morphosyntactic analysis, so called part-of-speech tagging. There exist several systems for morphosyntactic analysis of Swedish using different methods; handcrafted rules e.g. those found in the Swedish Constraint Grammar (Birn, 1998), and statistical approaches (Carlberger & Kann, 1999; Åström, 1996). Even though it is hard to compare the different evaluations made, they all achieve an accuracy of around 95%. This means that every 20$^{th}$ word is assigned the wrong tag. This has of course consequences for the later stage of grammar checking, which is built on this initial analysis.

Recent Swedish research on part-of-speech tagging has focused on smoothing methods (Nivre, 2000), ensemble methods (Megyesi, 2002a; Sjöbergh, 2003) and how the tag set size and training size matters for the performance of the tagging method (Megyesi, 2002a). Three parsers have been developed recently. Two of them use machine learning (Megyesi, 2002b; Nivre *et al.*, 2004) while the other is based on finite-state cascades, called Cass-Swe (Kokkinakis & Johansson-Kokkinakis, 1999). Furthermore, two commercial parsers also exist, developed in the frameworks of Constraint Grammar (Birn, 1998) and Functional Dependency Grammar (Voutilainen, 2001) respectively. The Functional Dependency Grammar parser actually builds a connected tree structure.

## 3.4 The grammar checker Granska

Central in this thesis is the grammar checker Granska. Although it is carefully described in Paper 1 in this thesis, it will be briefly described below. Also, some decisions made regarding its design are important to elaborate on here.

Due to the limitations of other approaches we decided to base Granska's general language analysis on part-of-speech tagging. At that time, most parsers described above were not developed, and one design decision was therefore to build a partial and shallow parser inside Granska. This later evolved to GTA. Off-the-shelf taggers existed (Brill, 1995), but they were not considered fast enough for a grammar checker to be used in an interactive mode. Faster part-of-speech taggers arrived with the TnT-tagger (Brants, 2000) and faster implementations of Brill's tagger, but that was some years after the Granska project had started. Instead, the research group decided to develop their own tagger, called the Granska tagger, which was seamlessly integrated with a specially designed rule language, and a rule matcher. The rule language and the rule development have been described in detail in (Knutsson, 2001).

### 3.4.1 A short description of Granska

The modular structure of Granska is presented in Figure 4. The user's text is first processed in the tokenizer. This module divides the text's sequence of characters into sentences and words. The sentence is then used as the unit for analysis. In the next step, a part-of-speech tagger is used to assign disambiguated part-of-speech and inflectional form information to each word. The rule matcher then applies error rules and general rules for phrase analysis and clause boundary detection on the tagged text. The error rule component can get correction proposals from a word inflector as well as using the general phrase analysis and clause boundary detection as contextual information in the error rules. Granska also includes the spelling checker Stava (Kann et al., 2001). This design solution makes it possible to use the general language analyzers in spelling checking, and to use Stava for checking split compounds, which have been compounded to a new word by the error rules. In a last step before presenting the result to the user, the rule matcher applies all rules again on the sentence with the results of the correction, in order to determine if another error was introduced by the correction proposal. In such cases, the correction alternative is discarded. Finally, the errors that Granska have detected in the text are pointed out to the user in a graphical user interface. All errors are diagnosed, and the program provides correction proposals on most of them.

Granska includes a statistical part-of-speech tagger (Carlberger & Kann, 1999), called the Granska tagger. This tagger is based on a lexicon and statistics gathered from the Stockholm Umeå Corpus (Ejerhed et al., 1992). The error rules have been written in an object-oriented rule language constructed especially for Granska. About 350 rules have been manually constructed, and about half of these are error detection and

correction rules. The rest of the rules are used for phrase analysis, clause boundary detection and to accept constructions that are quite similar to errors.

An example of a rule written in Granska's rule language detecting agreement errors in a noun phrase is the following. The rule will find an error such as "Jag såg **en gula bilen**" (Eng. I saw a yellow car) and propose a correction to "Jag såg den gula bilen" (Eng. I saw the yellow car").

```
cong22@disagreement
{
      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)
}
```

The first part of the rule detects the agreement error. The second part tells what should happen after a matching. The **`mark`** statement specifies that the erroneous phrase should be marked in the text, the **`corr`** statement that a function is used to generate a new inflection of the article from the lexicon, one that agrees with the noun. This correction suggestion is presented to the user together with a diagnostic comment (in the **`info`** statement) describing the error.

The rules are compiled and optimized using statistics of words and tag bigrams in Swedish. This means that each rule is checked by the matcher only at the positions in the text where the words or tag bigrams of the least probable position in the rule occur. This design allows a fast rule matching and relieves the grammarian from the constraint to write very few and efficient rules.

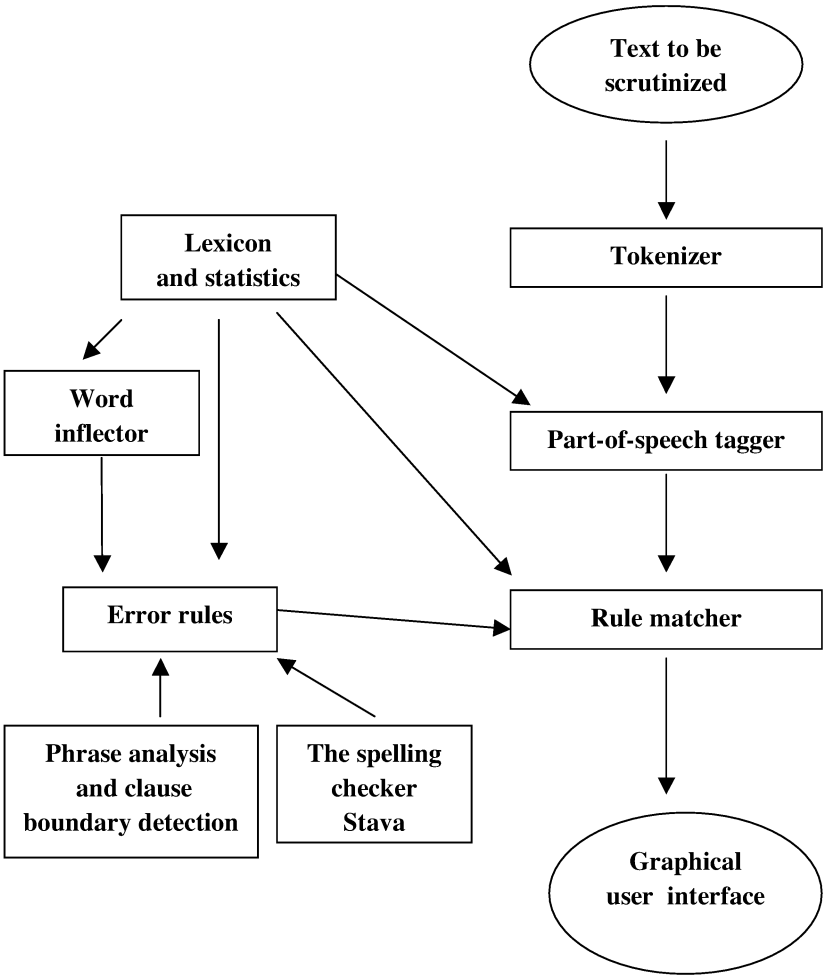


Figure 4. *An overview of the Granska system.*

### 3.4.2 What kinds of errors to focus on

When developing a phenomena-based grammar checker like Granska an error catalog has to be defined. This error catalog is first of all constrained by the limitations of the current technology. With this technology constraint as a point of departure the catalog can be based on:

- Frequent errors as pointed out by teachers or other experts.
- Books or lists of errors.

- Possible correct constructions that can be identified and their counterparts.
- Error corpora, both of native and second language writers' texts.

Granska is based on all four of the sources, and the core of Granska's error catalog is presented in Table 1. If there would have existed larger amounts of annotated errors in different kinds of corpora, Granska might have focused on other types of errors.

Table 1. *Granska's error catalog with examples of targeted errors.*

| **Error type** | **Example** | **Carefully developed in Granska** |
|---|---|---|
| Agreement NP | Jag såg **en glador** (Eng. I saw a kites) | Yes |
| Predicative | **Mannen** var **glada**. (Eng. The man was happy) | Yes |
| Split compounds | Gladan tog en **åker sork**. (Eng. The kite caught a field vole) | Yes |
| Verb chains | Jag **har spana** på glador en längre tid. (Eng. I have watch kites for a long time) | Yes |
| Miscellaneous | Jag visade gladan **för hon**. (Eng. I showed she the kite) | Yes |
| Spelling | Jag såg en **galda**. (Eng. I saw a kiet) | Yes |
| Missing X | **Jag en glada**. (Eng. I a kite) | Limited |
| Word order | Jag stannade hemma eftersom **glador flyger inte** i detta väder. (Eng. I stayed home because kites do fly not in this weather) | Limited |

As pointed out in Paper 1, the focus in the development of the error rules in Granska has been on three error types: agreement errors in noun phrases (NP), agreement errors in predicative and split compounds. Agreement errors within the NP and split compounds are frequent errors in Swedish texts, and they have also been studied by Domeij et al (1999).

Agreement errors in the predicative were chosen as they challenge the types of errors that are possible to detect with the methods used in Granska (Knutsson, 2002). In order to detect this type of error, almost the whole clause must be analyzed and disambiguated on the phrase level. However, Granska targets several other errors, such as missing subject or finite verb (Missing X), verb chain errors, and a group of minor error types such as subject form after preposition (Miscellaneous). Word order errors are very complex, and in Granska there are only a few rules, which try to detect the wrong placement of adverbs in subordinated clauses.

### 3.4.3 The partial and shallow parser GTA

A subset of the rules of Granska constitutes a starting point for a partial and shallow parser for Swedish, called GTA (Knutsson et al. 2003). Inside Granska the rules for phrase identification are used in many of the error detection rules. The rules became a "stand-alone" parser with improved disambiguation when ProbGranska was developed, and plays an important part in its error detection algorithm, see Paper 4. The parser has been carefully evaluated in Paper 3. It is also used to identify grammatical categories in Grim, see Paper 5 and Chapter 6.

GTA is rule-based and relies on hand-crafted rules written in a formalism with a context-free backbone, i.e Granska's rule language (Knutsson, 2001). The rules are augmented with features. It is quite often claimed that the grammars of shallow parsers are quite large, containing thousands of rules (Hammerton *et al.*, 2002). This is not the case with GTA. In total GTA contains 260 rules. 200 of these rules identify different kinds of phrases, 40 rules are disambiguation rules that select heuristically between ambiguous phrase identifications. Clause boundaries are identified with 20 rules quite similar to Ejerhed's algorithm for clause boundary detection (Ejerhed, 1999). However, the number of rules is not the only aspect of grammar complexity. Interaction between rules and recursion are also important.

The rules in the grammar are applied on part-of-speech tagged text, either from an integrated tagger (Carlberger & Kann, 1999) or from an external source. GTA identifies constituents and assigns phrase labels. However, no full trees with a top node are built. The disambiguation of phrase boundaries is in a first phase done within the rules, and secondly using heuristic selection. In a third phase, a disambiguation and selection algorithm called the Tetris algorithm is applied to the remaining ambiguities (Bigert, 2005). The analysis is surface-oriented and identifies many types of phrases in Swedish. The basic phrase types are adverb phrases (ADVP), adjective phrases (AP), infinitive verb phrases (INFP),

noun phrases (NP), prepositional phrases (PP) and limited verb phrases and verb chains (VC). The internal structure of the phrases is parsed when appropriate and the heads of the phrases are identified.

The identification of the head of the phrases are used as transformation rules in ProbGranska, e.g. a phrase like “den glada mannen” (Eng. the happy man) is transformed to its head “mannen” (Eng. the man). The clause boundary detection is also important in ProbGranska, defining the unit of error analysis (Bigert & Knutsson, 2002). The same rules are also used in the detection of for example split compounds in Granska.

# 4 Using language tools in second language writing and learning

Using language tools to support second language writing and learning is not a new area of practice or research. Writing aids like Writer's workbench have been used for more than 20 years (Reid et al., 1983), and more advanced grammar checkers (Brock, 1993) for about fifteen years. With the advent of widespread network technologies and the Internet, students are not only using word processors for composition; they are using computers when writing in computer assisted classroom discussions, Internet Relay Chat, and e-mail environments (Matsuda et al., 2003). Second language writing is a field on its own, but two important and related fields are Computer Assisted Language Learning (CALL) and Second Language Learning/Acquisition (SLL/SLA[2]). It is also a field where writing research meets applied linguistics, and where the insight seems to be that both perspectives are necessary in order to study second language writing (Matsuda et al., 2003).

The individual process-oriented view of writing has been criticized for not taking into account how language is used to construct meaning in sociocultural contexts (Hyland, 2003). Language is just not only a part of the editing phase of writing as in process oriented models of writing (Hyland, 2003). Language is central in writing. How to analyze and interpret a genre where the written communication should fit is constrained by social and cultural factors. These sociocultural factors, which are necessary to know in order to construct meaning in written communication, have to be learned by second language writers.

## 4.1 Language tools in computer assisted language learning

Computer assisted language learning (CALL) is a broad field, including many different theoretical and technical approaches to language learning

[2] The terms Second Language Learning (SLL) and Second Language Acquisition (SLA) are used more or less as synonyms in this thesis. However, they are grounded in different traditions with different theoretical backgrounds. The acronym SLA is more frequently used.

(Chapelle, 2001; Levy, 1997). The activities that are to be supported are hearing, speaking, reading and writing. The main component in CALL is that the computer is used in some way to facilitate language learning. The computer can be used as a means for mediated communication, for instance between native speakers and learners using chat or e-mail. Common CALL applications also include functions for automated language exercises, games, digital audio, video and hypermedia. The motivation for using CALL is according to (Nerbonne, 2002) especially strong whenever teachers are unavailable, such as in distance learning but also in regular teaching when there is not enough time and energy to help all students. There is also a great need for CALL outside schools and universities, for self-studies, in order to maintain language competence.

This individual ideology of learning has been criticized, pointing at the fact that how to use language in a genre is in many cases based on a intuitive basis, which in second language writing is very problematic when the writers have different social and cultural backgrounds (Hyland, 2003). The teacher is very important to aid the learner in understanding how different texts have different purpose, audience and message.

Language skills can also be supported in daily activities (work, education and leisure/social life) not necessarily including teaching, for instance special dictionaries for a second language or a grammar checker in a word processor designed for second language writers.

The degree of development of computer technology has always had a strong influence on which learning theories that have been used in CALL (Levy, 1997). However, technology has not been working on its own, theoretical findings about language learning have had a strong influence on the design of CALL activities.

## 4.2 Language technology in computer assisted language-learning systems

Most computer assisted language learning systems that are in use include no “intelligence” (Borin, 2002). However, in the following we will focus on applications where computer programs take a limited but active part in the process of language learning. Applications with strong focus on the control of pedagogy, so called intelligent language tutors will be left out in the following. Instead, focus is mostly on the feedback that a computer program can give on unconstrained language. Feedback can be given on the learner’s own production or on some other language material and media. One technology that has a potential for a task of this kind is language technology. Language technology or computational linguistics

has been only scarcely treated in mainstream CALL (Borin, 2002). In Levy (1997) and Chapelle (2001) computational linguistics seems to concern mostly syntactic parsing; this is a very limited scope of this interdisciplinary field. Instead language technology is a broad field, which can support different levels of language and different processes in language learning.

Nerbonne (2002) presents a good overview on how language technology is used today in CALL and how it can be used in the future. The following are some examples on applications based on language technology useful in CALL. Morphological analysers and generators can support both reading and writing. Concordances aid in advanced searches in large corpora, and for the exploration of bilingual texts. Dictionaries can be more accessible with a lemmatizer especially for language with strong inflection systems. Syntactic analysers can aid in error diagnoses or to visualize syntactic constructions in the target language. Language generators can help in text construction or reformulation. Pronunciation training can be done using speech recognition. Speech synthesis can be used for hearing exercises.

Many more applications are possible, and one main factor for the slow integration of language technology in CALL is the strong dividing line between the two disciplines (Borin, 2002). Nerbonne (2002) points out another important issue: there already exist successful CALL applications, so why bother about language technology with the uncertainty of these "intelligent" programs? Programs based on language technology are not perfect, and will not be in the near future. However, as Nerbonne argues, other tools like grammar books and dictionaries are not very well fitted to the learner's situation and activity. Language technology can provide an interactive link into this kind of "knowledge". Language technology can also help when the learner explores natural language data, or when teachers and learners collect relevant learning material based on these data.

Evaluating applications is an important part of research in CALL. But even more important is to evaluate activities rather than just the application, since it is what teachers and learners do with the application that matters. According to Chapelle (2001) a program or activity has high language learning potential if it involves the pedagogy focus on form (see below). Not all researchers in second language learning agree that this pedagogy increases the level of learning, and hence the evaluation of CALL systems in Chapelle's way cannot be done without a theory or method based in second language learning research or other relevant

disciplines. Chapelle's conclusion on the choice of pedagogy is based on her interpretation of current research results in second language research, but Chapelle also argues that what is considered to be the most suitable language learning pedagogy might change as research results change. To conclude, CALL must be based on research on second language learning.

## 4.3 Second language learning

An important issue in second language acquisition/learning research is the question about whether language learning is something different than other learning processes. The term acquisition indicates that many believe that. These beliefs have their origins in how the first language is acquired according to Chomsky's theories. Learning a language is said to be an unconscious process. Many researchers use the term *language acquisition* for acquired knowledge/skills on how to use language and *language learning* for learned knowledge about language (Ellis, 1997).

Within second language learning research there seems to be a consensus that behaviorist theories cannot explain second language learning; the second language learner does not simply reproduce the language from input. One example is that the errors made by second language learners quite often violate many constraints of the target language and without any resemblance of target language input. The second language learners seem to have their own linguistic system with its own rules – an interlanguage. The term interlanguage comes from Selinker (Selinker, 1972) and is based on the mentalist theory on first language acquisition. An interlanguage has its own grammar, and can be close to the target language on some forms, but have a gap or a hole on other forms. Whatever one thinks of the theoretical grounding of interlanguage, it is a useful metaphor when discussing second language development.

A theory that has had a strong influence on SLA is Krashen's *Input Hypothesis* (Krashen, 1982, 1994) in which the main component in language acquisition is receiving comprehensive input. According to this theory, reading and hearing comprehensively the target language is enough for learning a second language. A learner goes from one language level to another by understanding the message of the above level using the context and extra-linguistic information. In this way the interlanguage is developed. Chomsky's language acquisition device (Chomsky, 1965) is important in the input hypothesis, partly because of the "complexity argument" – all rules and words cannot possibly be learned consciously. Krashen strongly argues against all hypotheses that emphasize the importance of output and feedback on output, and one argument is that many learners become very good language users with very little training

and practice in language output (Krashen, 1994). Krashen's conclusion is that the effect of oral and written output is minimal for the acquisition of a second language in educational environments.

As with many other strong hypotheses there is sooner or later a reaction to the proposed hypothesis. In second language learning this reaction emphasizes a focus on interaction, output and feedback. Long proposes an interaction hypothesis (Long, 1981) and Swain formulates what she calls "the output hypothesis" (Swain, 2000). Long has observed that the most common way to make input comprehensive, is not to make input itself more comprehensive, instead it is common to adapt the interaction through negotiation to make input comprehensive (negotiation of meaning). In other words, input gets comprehensive through communication.

Swain puts forward that hypothesis testing (i.e. the learners test their hypotheses about how the target language works) is important in second language learning, based on the "noticing the gap principle", and more simply put: hypothesis testing gives the learners the opportunity to use language. According to Swain the focus on output sets the learner in control, and in speaking or writing, learners are pushed to use language more deeply, not only focusing on semantic aspects of language.

These perspectives focus on learner interaction and communication, but influential researchers like Long and Swain also work with a complementary concept called "focus on form". Focus on form (Long & Robinson, 1998) is based on the insight that communication is not a panacea – in order to use language in a target-like fashion form matters. Focus on form means to draw the learner's attention to linguistic code features while conducting meaningful tasks (i.e. achieving communicational goals). What kind of form to focus on and what kind of feedback that is most appropriate is a question for debate (Doughty & Williams, 1998), see also Chapter 6 in this thesis.

Studies focusing on form draw the attention to important issues in second language learning like how the interlanguage should keep developing, how "holes" in the interlanguage should be made conscious to the language learner, and how the gap between the interlanguage and the target language should be made salient to learners. Many researchers have emphasized that language consciousness and meta-linguistic reflection are fruitful for comprehensive and dynamic interlanguage development (Lindberg & Skeppstedt, 2000). Focus on form also pays

attention both to input and output, but many questions are still open, see Doughty & Williams (1998).

A perspective that has reached an increased attention in recent years is sociocultural approaches to learning. This field has its origins from the works of Vygotsky (Lantolf, 2000), but important extensions of the theory has been made (Wertsch, 1991). These extensions also have impact on theories of second language learning. From a sociocultural perspective, learning a language is much more than "cracking the code" (Lindberg, 2001). Rather, learning a new language also includes to learn and understand the sociocultural setting in which the language is spoken and written. Second language learning is from this perspective based on social interaction, where the learner is interacting with people through language mediated by social and cultural contexts, and different kinds of tools, psychological as well as technical.

It is currently hard to see any alternatives to focus on form without returning back to pure communicative approaches, which according to many leading researchers within the field of second language learning is not enough in order to learn a language. Furthermore, sociocultural studies of grammatical form give us something different (DiCamilla & Lantolf, 1994; Lantolf *et al.*, 1997). In these studies grammatical features are seen as psychological tools for organizing and guiding mental activity. If we connect this tool metaphor with focus on form, the concept gets another grounding; it provides us means to visualize the tools also called grammatical forms, see also Nunn (2001).

# 5 Evaluating language tools

Computer programs including language technology have mostly been evaluated with narrowly defined tasks inside the laboratory, and recently critical voices have been heard against the studies of systems and not the systems' roles in human activity (Sparc Jones, 2001). The measurements used in these evaluations are good when testing and comparing the performance of different algorithms, but it is very unclear what they tell us about the systems' performance when users are using the systems in different activities (Karlgren, 2000).

What kinds of programs that are to be evaluated is also important. There is no need for including users when for instance evaluating the syntactical parser used as one module in a grammar checker. But when it comes to the study of the use of the grammar checker the users' activities in realistic settings are of utmost importance. Thus, two different approaches are necessary to apply: technical evaluations using different evaluation corpora and user studies to get an understanding of the role of language technology in the human activity, e.g. writing a text.

The two fields language technology and human-computer interaction have as one of several goals to use evaluations as a means to improve the design of the systems. The outcome of the different evaluations in many cases leads to proposals for a new design or redesign of the current systems. The following studies all have such an aim to inform the core algorithms as well as the design of user interface and interaction.

When it comes to the evaluation of a grammar checker it is necessary to evaluate the programs' performance on texts with errors in them. The processes that should be focused on are the program's capacity of detecting, diagnosing and making correction proposals on texts. These three subprocesses are of different complexity, and should also be treated as three different evaluation tasks. A program can still be useful if the error detection capacity is good but the ability to diagnose the errors is low. This depends on what kind of feedback the user wants or needs in a specific activity.

Two evaluations that are based on Granska's performance on the product, the text, are presented in Paper 1 and Paper 2. The first evaluation is based on 201 019 words from five different genres. The second evaluation is smaller, but the texts are written by second language learners and collected within the field study presented in Paper 5. In Paper 4 there was also a textual evaluation made, but not by me. It also uses text written by second language learners, and can be used as a comparison to my evaluation on second language learners' text in Paper 1.

We also need to know how the general text analyzers used inside the grammar checker perform on texts with errors in them. The error detection algorithm relies heavily on the output from the general analyzers' capacity to analyze; hence this step is critical for the performance of a grammar checker. It is hard to see that the checking module will find many of the errors "hidden" by the general analyzers, like part-of-speech taggers and parsers. An evaluation focusing on this issue is presented in Paper 3. The errors that violate the textual norm are spelling errors introduced in the texts.

Evaluations of the design of the user interface and interaction are important parts of the user studies. A grammar checker can in many cases return a lot of linguistic information as output. However, only a limited amount of this information is relevant for the user at a certain time of interaction with the system. What kind of information should be presented and what kind of meta-language that should be used are important questions to study. The grammar checker engine can also be used in different modes of interaction with the user. Which mode or which modes that are preferable for the users are central questions to study. These questions have been studied in the formative user study in Paper 2, and in the field study in Paper 5.

When studying writers and learners, the understanding of the context, for example the learning activity or pedagogy is also very relevant to get hold on, and this lies mainly outside the computer environment. A grammar checker can be used in different ways, and with different levels of success depending on how the users are instructed. A novel writer, for instance, might need to learn how to use the feedback from the grammar checker through a teacher. These questions played an important role in the study presented in Paper 5.

## 5.1 Textual evaluations of language tools

An important part of developing language tools in research environments is to have the possibility to make different evaluations of the tools. Without a detailed understanding of the different algorithms and linguistic databases used, the evaluation might be shallow, resulting in limited insights about the causes of the "behavior" of the language tool under investigation. In addition, the possibility to change different options in the programs is also very important when conducting evaluations.

Granska and related technology was developed to be used in evaluations. To achieve this, a language engineering perspective was necessary to make the language tools robust enough for realistic and large scaled evaluations.

The growing interest in statistical and empirical methods in computational linguistics based on different kinds of corpora has made the evaluation procedure an obvious and important part of research. In several sub disciplines annual events have evaluation as main focus, e.g. TREC (information retrieval), MUC (information extraction), DUC (text summarization), and for Europe CLEF (cross linguistic information retrieval). But research on grammar checkers and other writing tools has been lacking such an event. However, the same evaluation procedures can be used to some extent, but of course reinterpreted in the context of grammar checking and writing tools. Recall and precision are the well-established measurements in many evaluation procedures, e.g. information retrieval (Karlgren, 2000) and are also applied in the evaluations of grammar checkers.

## 5.2 What is important to measure?

An obvious and most relevant question posed by many people is: How many errors will the program find? This question is addressed by the measurement called recall. However, in order to measure recall we need to find all errors in texts that give a broad picture of language use. This is much harder than it might look for several reasons. To start with, a representative text collection is not easy to define. In addition, what types of errors should the program find? Another obvious question posed is how many false alarms will a user get using your program? This is measured with the measurement called precision.

Using the measurements recall and precision all error types are normally treated as having the same value, which can make a system look much

better than it is. For a writer it might be more important to find word order errors that change the meaning of a sentence than for instance detecting a verb chain error.

Precision is easier to measure than recall. The evaluator has only to scrutinize all alarms in the texts; all errors do not have to be identified. To mitigate the problem of recall, we have in some of the evaluations used something called maximal recall, which can only be used when two or more grammar checkers are evaluated on the same texts. When using maximal recall the errors found by the different grammar checkers are considered to be all errors that exist.

### 5.2.1 Defining recall and precision

In evaluation terminology the variables true/false positives and true/false negatives, are commonly used. These variables can be used in many areas where a system that tries to detect different things should be evaluated, like for instance a grammar checker searching for instances of errors. If a system does something correctly it is true, otherwise false.

Looking for errors in texts can be compared with looking for mushrooms in the woods. There are a lot of mushrooms in the woods, and some are edible and some are not. We want to find the edible mushrooms, which in the textual case would mean the errors. We call these for true positives. The non-edible mushrooms that we found but which we believe are edible are called false positives. All edible mushrooms that we do not find or ignore because we are not sure are called false negatives. The non-edible mushrooms that we ignore are called true negatives. In Table 2 the distinctions are translated to the case of a grammar checker.

Table 2. *True/false positives and negatives used in context of grammar checking. The table is adapted from (Manning & Schütze, 1999).*

| **Grammar checker** | **Actual** | |
|---|---|---|
| | **target (language errors)** | **not target (grammatical language)** |
| detected | *true positives* (*tp*) are correct detections of errors. | *false positives* (*fp*) are false detections also called false alarms. |
| not detected | *false negatives* (*fn*) are missed errors. | *true negatives* (*tn*) correct misses of grammatical language. |

True positives and true negatives are the detections and non-detections respectively that are made correctly by the grammar checker. True positives are correctly identified errors and true negatives are bits of grammatical language that are correctly ignored by the grammar checker. False positives are bits of grammatical language that are signaled as errors, these are also called false alarms. False negatives are real errors that have not been detected as errors by the grammar checker. The numbers of each variable are counts of detections, possible detections and non-detections. These variables are used in the calculations of precision and recall. A combination of the two can be made using a formula called F-score[3], where β is used as a weight either for recall or precision. When β is greater than 1, precision is favored, and when β is less than 1, recall is favored. β = 1 is normally used, meaning that recall and precision is equally weighted.

Precision def: $P = \frac{tp}{tp + fp}$

Recall def: $R = \frac{tp}{tp + fn}$

F-score def: $F = \frac{\left(1 + \beta^2\right)PR}{\beta^2 P + R}$

[3] The term F-measure is also used for the same measurement.

## 5.3 Corpora used in the evaluations of Granska and related programs

Corpora can be used for various things. In language technology they are often used as training material for the development of different algorithms, and for the evaluation of these algorithms. In this thesis one corpus has played significant role in both development and evaluation: the Stockholm-Umeå Corpus (Ejerhed et al., 1992), also called SUC.

The evaluation of Granska presented in Paper 1 and Paper 2 was based on a compilation of different texts from various sources, forming five genres of 201 019 words. The text genres were sport news, international news[4], public authority text, popular science and student essays. The texts from international and sport news were taken from the KTH News Corpus (Hassel, 2001). To use different genres for evaluation is essential in order to get an understanding on the program's performance on different language use. In this evaluation all errors were manually searched for, which made it possible to measure both recall and precision. More details on this evaluation is also found in (Knutsson, 2001).

The evaluation of Granska on second language learners' texts presented in Paper 1 was made on 32 452 words from the CrossCheck corpus[5] (Lindberg & Eriksson, 2004). To manually search and analyze all errors in these texts written by second language learners is a complicated task, and therefore maximal recall and precision were measured. The evaluation made in Paper 4 was carried out on 10 000 words from the SSM-part in the CrossCheck corpus. The CrossCheck corpus has not been manually annotated with error tags.

The SUC corpus has been used in the evaluations carried out in Paper 3. In this evaluation, about 15 000 words from SUC were annotated with basic phrase tags and clause boundary tags, forming a small and flat treebank. To mitigate the lack of annotated Swedish error corpora, we have used artificial errors created by error generating rules. Artificial errors were automatically introduced in the annotated texts from SUC. In this procedure, errors can also be automatically annotated, since we know which error types the program is generating. The annotated resource with artificial errors was used in the experiments carried out on tagger and parser robustness against spelling errors as presented in Paper 3.

---

[5] The CrossCheck corpus has also been denoted the SVANTE corpus.

## 5.4 Studying the use of language tools

Evaluating language tools in the context of writing and learning is a complex task. Studying writing must involve the study of the writing activity and not only the written text.

Trying to answer the question if language tools are good or bad for the resulting text is very hard, but more problematic is that this question only focuses on the result, the final text, and not the writing processes. On the one hand, if this is the only question to study, we both view the writer and the language tools as black boxes. On the other hand, the question is relevant if it is combined with questions focusing on the process and writing context. Dicamilla and Lantolf (1994) argue for the importance of the interplay between grammatical forms and cognitive functions. Thus, we need to study both the textual form and the language used for controlling writing. In addition, the process and the activity, the social and cultural context and the mediation of tools are necessary aspects to study in order to achieve an understanding of the role of language tools in writing. The big question is then: how do language tools mediate such processes?
In the studies conducted in this thesis we will only touch upon these questions; the main aim of the user studies was to get an understanding of the role of language tools in writing and to contribute to the improvement of the design of existing tools for writing in writing and learning environments. The first objective was to improve the design of Granska. The second objective was to develop a new language environment based on design principles, which were identified during field studies with Granska. However, the view of writing and learning a second language through writing activities is very important in the design of studies focusing on writing.

### 5.4.1 Writing processes are difficult to access

Writing processes have been studied from several perspectives, based on different views of human mind and cognition. Domeij (2003) discusses three main perspectives: the text-oriented perspective with a focus on the product – the text; the cognitive perspective, with focus on the individual and internal representations (Flower & Hayes, 1981), and the third perspective is the ethnographical perspective, with a focus on writing in real contexts outside the laboratory. However, all studies carried out outside the laboratory cannot be called ethnography. Instead, ethnography is one of several ways of conducting field studies.

Ethnography focuses on skills and practices conducted by people during authentic activities, and the researcher's role is to study the setting, and the participants and interact with them in order to get an understanding of the activity. This research in not driven by hypotheses, and the result is a descriptive written text (Normark, 2002). Studies within the paradigm of sociocultural theory are also focusing on authentic activities (Lantolf, 2000), but the activities are approached with theoretical grounding in the works of Vygotsky, Luria, Leont'ev and Bakhtin and their followers (Wertsch, 1991). This kind of research is based on theory-guided observation and interpretation of people involved in different kinds of activities (Lantolf, 2000).

Within the cognitive perspective, think-aloud protocols have been a quite common method to acquire knowledge about the writing process (Smagorinsky, 1994). The method is based on the collection of verbal data: the researcher instructs the participants to talk about their own writing process while writing. Some studies with special focus on the role of language tools on the revision process have also employed thinking aloud protocols (Domeij, 2003). Think-aloud protocols have been criticized by several scholars, for example researchers within the sociocultural theory tradition (Säljö, 2000). The main argument against the method from a sociocultural perspective is that the "thinking processes" that the think-aloud method makes accessible are constrained by rules of communication. The utterances are produced in order to be understandable for the researcher and the information the researcher gets is the participants' analyses of their own thinking (Säljö, 2000).

While think-aloud methodologies are focusing on verbal data, methods based on logging collects what the users are doing with the computers by collecting for instance the users' keystrokes, mouse movements and pauses using special computer logging programs (Kollberg, 1998). An attractive thing with logging methods is that the writers will hardly notice that they are studied. One problem with the method might be that the researcher gets the result of the writer's decision when composing text, and not the decision itself, which is the one that the think-aloud protocol tries to capture.

Much work in this thesis has focused on errors, and that is also the case in the two user studies conducted. The focus on errors is not only motivated by the study of a grammar checker; there are several other good reasons for this perspective. Errors are part of the writers and learners' writing practices, and can be viewed as the expression of a conflict between the between the writer's conceptions of what is correct use of the written

language and what is really correct use of the written language. The studies are not focusing on the origins of the errors, neither their psycholinguistic interpretation, instead focus lies on how to support and give feedback on the errors that the grammar checker can detect.

## 5.5 Two different user studies with Granska

In the papers, which this thesis is based on two user studies are presented. The first study was a so-called formative study (Paper 2) and the second study was a field study (Paper 5). The two studies are different, although one question is still the same: how do the users deal with the feedback from Granska? The main difference between the studies is the context in which the studies were carried out. In the first study the users worked with an already written text, but in the second study the participants used only their own texts when interacting with Granska. The time variable is also very different; the first study took about two hours per user, whereas in the second study the users used Granska during two to four months of time. The first study focused on a kind of "traditional" use of a grammar checker, whereas the second aimed to study the use of a grammar checker in learning environments.

### 5.5.1 A formative user study

The first study was conducted inside the laboratory. Five users participated in the study and they were all experienced writers and had all to some extent, used spelling and grammar checkers before.

The questions raised in the study concerned both the core of the grammar checker and the user interface design. The grammar checker in MS Word and the grammar checker Granska were used in the study. The study focused on false alarms, wrong diagnoses and multiple suggestions from the programs. These issues all address the trade-off between recall and precision. If the program is designed for high recall false alarms, wrong diagnoses and multiple conflicting suggestions will be the result. If high precision is the main design objective a lot of alarms will be missed and alternative interpretations of an error will not be given. Results suggest that several conflicting diagnoses and proposals seem to be a limited problem, and that false alarms were to a variable extent difficult for the users to handle. A concrete design decision from this study was to continue with multiple diagnoses on the errors, and leave the decision of which alternative that is correct to the user.

### 5.5.2 A field study

While the former study was conducted inside the laboratory we wanted in this second study to get out into a more realistic context. When it comes to learning through writing, and second language learning in particular, we cannot isolate the process of learning when we view the learner as being part of a whole (van Lier, 2000). The learning environment is the social world.

We chose an advanced university course in Swedish as second language as the context for our study. A major concern was to study the use of Granska in learners' free writing. This raised several implications for the design of the study. First of all the tool has to be introduced in this new context. The first step was to establish and develop a relationship with the teacher, which was necessary in order to study the learners' use of Granska in connection to the teaching.

The next step was to gain the learners' confidence and make them participate in our study on a voluntary basis, outside class. In a first meeting at the university we carefully explained the tool Granska and emphasized that Granska is a computer tool with limited knowledge of language. Those who wanted to participate were encouraged to use Granska outside class and the instruction was the following "use Granska whenever you want and when you feel it will help you".

The next question was: how should we study the use of Granska both in the university computer labs as well as in the homes of the participants? We chose a web interface to Granska as it made it possible for the participants to use Granska in different computer environments. The participants' texts were important and we collected two versions of each text; one before the use of Granska and one after the use. The idea was to compare the two texts in order to study if the participants have followed the advices from Granska.

A judgment procedure was also developed to get detailed information on what the participants thought about Granska's feedback in form of error detections, diagnoses and correction proposals. We instructed them to judge every alarm from Granska on a scale from 1 (incomprehensible) to 5 (excellent). This method has a resemblance to the more well-known method called grammaticality judgment, frequently used in second language research (Gass, 1994). In order to judge and grade Granska's alarms the participants have to more or less judge the whole sentence's grammaticality, to find out if Granska's feedback is right or wrong.

In addition, we encouraged the participants to write comments about all their interactions with Granska. All this material was sent to us by e-mail. These correspondences were also a good way to keep the contact with the participants and encourage them to continue to deliver data.

This way of collecting data depends a lot on the participants' willingness to take part in the study, which results in different amounts of data from every participant. Some of the control of the data collection is thus left to the participants. We used direct observations, pre-questionnaires, and post-interviews as complementary methods for data collection in the study.

The results from this study have been used in the design and development of Grim, which is presented in the next chapter.

# 6 Designing a language environment for second language writers of Swedish

The first seed of the design and development of a new language environment for writers of Swedish, called Grim, was the idea of visualization of grammatical categories in texts. The insight was achieved through user studies with Granska (Paper 5) and that feedback on errors provided by for example a grammar checker is not all that is needed in order to write correctly or target language-like.

First of all, the ideal grammar checker is out of scope of current technology and it is in fact hard to see its birth. Although current technology is quite robust it is also to a great extent superficial. In addition, there are many other skills that must be developed in order to learn a second language. Other tools might fill some gaps of the grammar checker, but indeed not all. However, during the user studies as presented in Paper 5, we found that a second-language learning environment should at least support:

- Focus on form, providing the learner with explicit lexical and grammatical feedback.
- Focus on authentic language use, providing the learner with implicit linguistic knowledge.
- Language exploration and reflection, for instance by visualizing grammatical categories and concordances.
- Different and competing views of the learner's interlanguage and the target language.

Grim builds on Granska, but also integrates other tools into a flexible user interface. The functionality in Grim is based on independent programs running on different servers; the program on the user's computer is just a client, with basic word processing functionality, and with interactive user interfaces to the independent programs to give immediate feedback. However, for the users, Grim appears as one single program with a

seamless integration of language tools. The users can work with their own texts as well as text on the target language.

## 6.1 Grim and focus on form

The understanding that the grammatical forms are complex having both cognitive and communicative dimensions is important (DiCamilla & Lantolf, 1994). This is also manifested in the concept "focus on form" (Long & Robinson, 1998). Swain (Swain, 1998) argues for focus on form based on insights from her studies:

"More than 2 decades of research in French immersion classes suggests that immersion students are able to understand much of what they hear and read even at early grade levels. And, although they are well able to get their meaning across in their second language, even at intermediate and higher grade levels they often do so with nontargetlike morphology and syntax" (Swain, p.65, 1998).

In other words, you need to know how to spell words and produce more or less grammatical and acceptable sentences in order to communicate above the most basic communicative levels. The concept focus on form is established through several studies in classroom environments (Doughty & Williams, 1998). One important feature of focus on form in contradiction to focus on forms is that the learner's attention should be drawn to linguistic code features when the learner already is involved in meaningful tasks or communication. The traditional focus on forms is exercises with no focus on meaning or communication. In focus on form the learner's attention should be drawn to morphemes, words, collocations, grammatical categories and structure, agreement, anaphora, pragmatic patterns and so forth. Thus, focus on form is both forms and rules (Doughty & Williams, 1998).

In the field study presented in Paper 5, focus on form was quite central in the language course, which we were studying. The teacher scrutinized all texts written by students and gave them feedback on their errors. The teacher also discussed common and significant errors. Using the terms of Doughty and Williams (1998) this must be seen as reactive focus on form. When it comes to the design of a computer environment including language technology, a grammar checker can be viewed as supporting reactive focus on form – the learner has to produce an error in order to get some feedback. In the language environment Grim the grammar checkers used are Granska and ProbGranska, see Figure 5. Using these two

grammar checkers with two different perspectives on errors provides the user with the opportunity to compare their results.

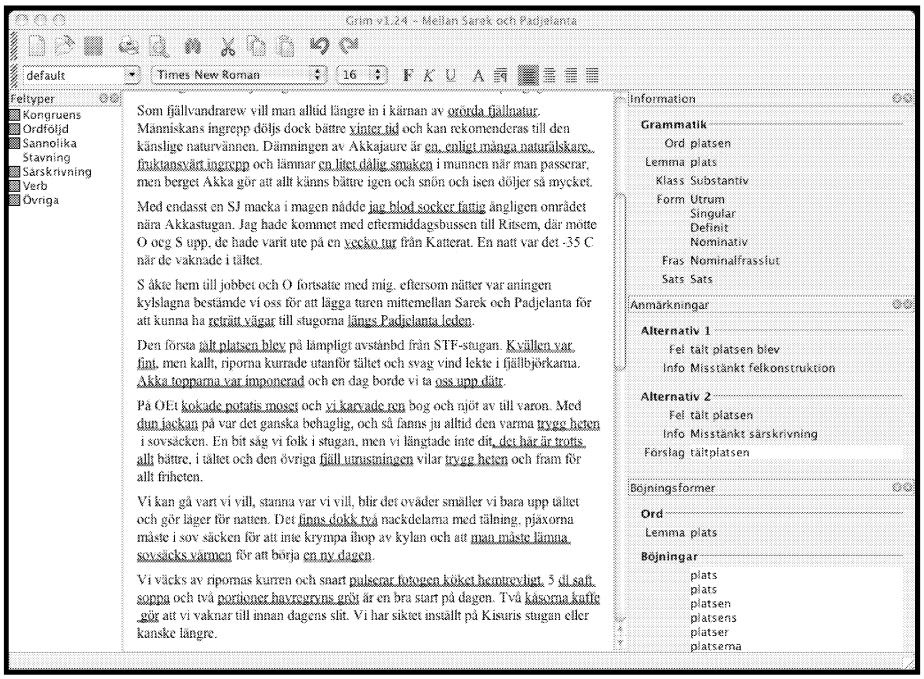


Figure 5. *Granska and ProbGranska in Grim. ProbGranska is hidden behind the error type "Sannolika" in the left column. On the split compound "tält platsen" Granska and ProbGranska agree, which is seen in the right column by the two diagnoses "Misstänkt felkonstruktion" (ProbGranska) and "Misstänkt särskrivning" (Granska). Only Granska can provide the user with the correction proposal "tältplatsen".*

A tool which is included in Grim is the partial and shallow parser GTA, and together with the part-of-speech tagger in Granska, these two tools can aid the user with what Doughty and Williams call "proactive" foucs on form". This more general text analysis on word classes and phrases in Grim can be used in a proactive manner – the teacher can instruct the learner's to highlight for instance all verbs and all noun phrases in the text analyzed in the language environment, see Figure 6.

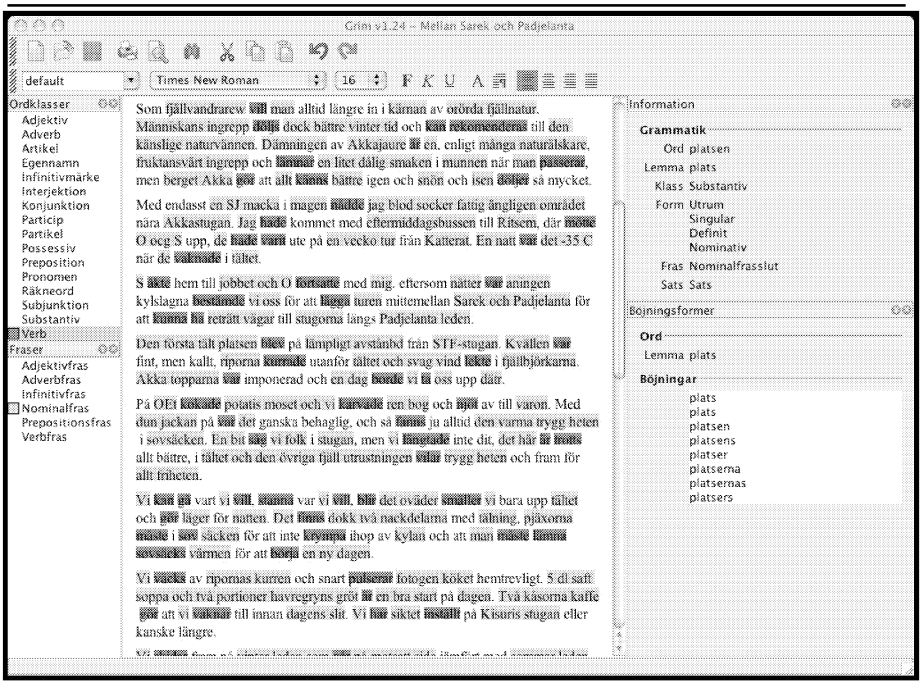


Figure 6. *All verbs and all simple noun phrases are highlighted in Grim. In the right information column linguistic information about the word "platsen" is presented.*

A delicate question is to choose the right form to focus on for the current group of learners. A smorgasbord of different forms to choose from is what a computer program ideally should be able to make salient for the learner. However, technology gives at the same time a lot of forms to focus on, but also strong limitations on what kinds of forms to focus on. Using all tags from the part-of-speech tagger in Grim, it is possible to provide the user with about 145 different forms. High-level forms like subject and object can only to a very limited extent be highlighted in Grim.

These possibilities and limitations of technology also raise design questions for the user interface, where too many alternatives, both of forms and rules, are hard to get an overview of. In addition, the forms, which can only be to a limited degree highlighted, might not have a natural place beside more full-fledged analyzed forms.

Focus on form is as already mentioned, both forms and rules. A flexible teaching strategy is using both explicit and implicit approaches, and that must also be the case in computerized learning environments. However,

the underlying technology might not give access to both forms and rules at the same time. An example pointed out by one of the participants in the field study in Paper 5 concerned the spelling checker. It can find errors (erroneous forms), but it can hardly "explain" the current spelling rule that has been violated, or at least, the spelling checkers have not been designed to output spelling rules. The participant wanted to learn the rule, not only the form.

## 6.2 The role of lexica and corpora in Grim

One design principle, which Grim is based on, is to provide the user with different and competing views on the learners' interlanguage and the target language. One view of the target language is mediated by different lexica. This view has in some cases a competing view on language based on information taken from corpora.

The spelling checker Stava for instance is based on the dictionary from the Swedish Academy version 11, which can be contrasted to the statistical processing by Granska's part-of-speech tagger, which has collected statistics on word forms and word form suffixes from Stockholm-Umeå Corpus. This has the effect that a word like "årsverke" (Eng. amount of work carried out by one person during one year) which is identified by Stava as a spelling error, will still get an analysis from Granska's part-of-speech tagger as a noun in neuter and singular form, which is correct. Another effect is that a noun phrase that contains a disagreement error like "en årsverke" will be highlighted as a disagreement error, but it will still be interpreted as a noun phrase. The correction proposals are also competing which can be seen in the right column in Figure 7 (Alternativ 1 vs. Alternativ 2).

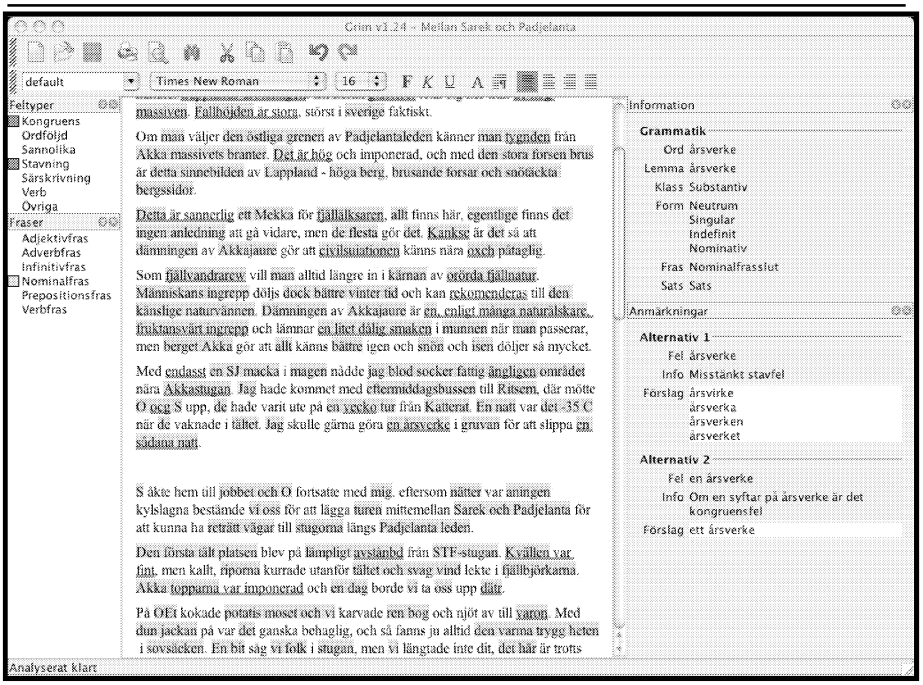


Figure 7. *Several diagnoses and correction proposals are quite often given as feedback to the user. The word "årsverke" has incorrectly been detected as a spelling error (Alternativ 1 in the right column). But, Granska has found an agreement error between the determiner "en" och "årsverke" (Alternativ 2 in the right column).*

So far, the language tools in Grim have provided an active and automatic view of language. However, two other tools in Grim are more static, but their use has been smoothed by language technology and user interface integration. The first tool is Lexin (2003) that is accessed by the user's click on a word in the current text, which is analyzed by Granska's tagger. The lemma form is then used to get more hits when searching the Lexin dictionary, which like many other dictionaries does not contain all inflections of a word.

The second tool can be used for language exploration of word use in authentic written language. This tool contains an interface to the Parole corpus (Gellerstam *et al.*, 2000), see Figure 8. The user can search for every word in her text, just by clicking on the current word, and choosing "Parole" from the tool-menu. No search expression has to be written, which is often the case in standard concordance expressions. If a user wants to search for all forms of a word, the same tool that is used to generate inflections for correction proposals in Granska is used.

Lexin and concordances in Parole provides complementary views of language, but can also be used together. Lexin can work as a tool to connect the learner's first language with the second language. Parole can be used to explore how target language forms provided by Lexin are used in authentic texts. Lexin translates "new" words that are made salient for the user through the concordances in the Parole interface.

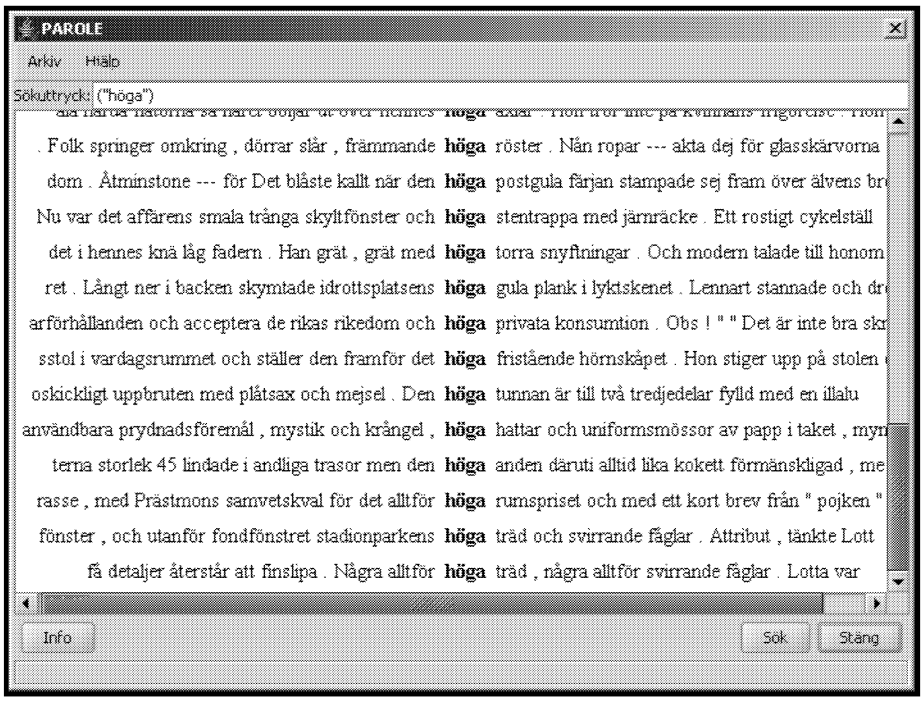


Figure 8. *Concordances on the word "höga" (Eng. high) made in the Parole corpus from the user interface in Grim.*

## 6.3 Preliminary findings and future work with Grim

No formal evaluation of Grim has yet been finished, but experiences so far suggest that Grim has new and interesting qualities as a writing and learning environment. According to Skeppstedt (2005) who made an informal study of Grim, it has potential for second language learning. Skeppstedt sees learning potential not only for advanced students, but also for learners at intermediate levels. She also points out that the program can be used for reading and not only for writing, and that the teacher must instruct and help the learners when they use Grim.

In a small formative user study with Grim results showed that the program's functions for explicit and implicit feedback complement each other (Pihl *et al.*, 2003). Results also suggest that the feedback from the different tools in Grim should be more integrated. A design proposal is suggested in which all possible feedback on a word is presented in the same view. The current version of Grim has moved towards this design, but the design proposal requires immediate feedback not only from Granska and GTA but also from all tools. In order to accomplish that, faster server solutions have to be built for all tools included in Grim.

In an ongoing user study in which students use Grim in a collaborative language learning task, a first preliminary result suggests that the learners focus to a great extent on the feedback on errors given in Grim, although they are aware of the limitations of the grammar checkers in Grim (Karlström & Cerratto Pargman, forthcoming). This result suggests that explicit feedback on errors has a strong influence on the learners' writing activities.

Even though Grim is a program with many users around the world, Grim's role in second language writing and learning must be further studied. The program opens up possibilities to use focus on form in teaching. However, user studies with teachers and students using Grim as a means in this pedagogy remain to be done. In addition, the language tools integrated in Grim, such as Granska has not been dramatically improved since the field study (Paper 5), and therefore more effort has to be put in these tools, and the way their feedback is presented to the user. The question on how to support the user's judgments of the feedback from the different tools in the program has to be seriously addressed in future studies with Grim.

# 7 Summary of the papers

In this chapter the included papers are summarized.

## 7.1 Paper 1: The development and performance of a grammar checker for Swedish: A language engineering perspective

### 7.1.1 Aims and background

Grammar checkers for English have been developed since the 1980s. For the Scandinavian languages, including Swedish, advanced tools have been lacking. In the late 90s several programs were built for grammar checking of Swedish.

In this paper a project with the aim to build a fast, robust and accurate grammar checker for Swedish called Granska is presented. In addition to the computational linguistic aspects of building a grammar checker, engineering aspects of such a project are also treated, as well as research with focus on interaction design of a tool like a grammar checker.

### 7.1.2 Methods

Granska combines statistical and rule-based methods. A statistical method is used in the general analysis of the words in the sentences; the general judgment of grammaticality is made by a statistical part-of-speech tagger based on a second order Hidden Markov Model. The statistics of tag unigrams, bigrams and trigrams and word-tag pairs are extracted from the SUC corpus.

Unknown words are treated by statistical means, based on the relative counts of word types ending with the same 1 to 5 letters. Compounds are analyzed by identifying the last word in the compound, and using its tag as the tag for the whole word. Words with an initial capital letter got an increased probability as proper nouns.

Phenomena based-rules are written to detect, diagnose and provide correction proposals on errors in text. This means that pre-selected error types are searched for with handcrafted rules written in a powerful rule

language. The rules are quite general, and have a special focus on three error types: split compounds, agreement errors within the noun phrase and agreement in predicative. However, other errors types are also searched for.

The rules are compiled and optimized using statistics of word and tag bigrams by a rule matcher. This means that the rule matcher checks the rules only at the positions in the text where the words or tag bigrams of the least probable position in the rule occur.

In the design of Granska we have integrated the spelling-checking program, Stava, into the grammar checker. By doing this, Stava can use Granska's information about the words when checking spelling, like proper names. The rules in Granska can also use Stava as a filter when checking for instance possible split compounds. Spelling corrections that cause errors according to Granska's grammar checking are stopped.

### 7.1.3 Evaluations

Two evaluations of Granska were carried out. The first evaluation of Granska was based on a compilation of different texts from various sources, forming five genres on totally 201 019 words. The text genres were sport news, international news, public authority text, popular science and student essays. To use different genres for evaluation is essential in order to get an understanding on the program's performance on different language use. In this evaluation all errors were manually searched for, which made it possible to measure both recall and precision. The overall recall on the five genres was 52% and the precision was 53%.

The second evaluation was made on 32 452 words from the CrossCheck corpus. The texts were written by advanced learners of Swedish. The spelling and grammar checker in Microsoft Word were used as a comparison to Granska. The grammar checker of MS Word has better precision, but much lower recall that Granska. For the spelling checkers, it is the opposite; the spelling checker of Word has a very high recall, but quite low precision, mostly due to bad performance on compounds and proper names. The spelling checker of Granska has quite a low recall, especially when it comes to erroneous compounds, which still seem to be a problem for Swedish spelling checkers.

### 7.1.4 Findings and conclusions

Granska is a fast grammar checker that has a good or better performance than comparable grammar checkers for Swedish. The rule language of Granska is also powerful enough to implement most of the rules from

other grammar checkers, and can therefore take advantage of good handcrafted and well-tested rules from other systems. The optimized rule matching used in Granska will make it possible to scale Granska to large amounts of rules. The integration of the spelling checker Stava in Granska might look like a simple design solution, but it is a fruitful way to improve spelling and grammar checkers.

## 7.2 Paper 2: Different ways of evaluating a Swedish grammar checker

### 7.2.1 Aims and background

The major aim of this paper is to demonstrate the strength of using several different methods when evaluating NLP-systems. This is especially important in the light of the heterogeneous group of users who are using writing tools based on NLP. One claim in the paper is that standard methods for evaluations within language technology based on recall and precision only address the issue of usability to a limited extent. The struggle and trade-off between recall and precision is a well-known problem in NLP, and there is no single answer what is preferable for most users. On top of that, aspects such as user abilities and needs, variability of writing task, text genre and user group, and the complexity of error types and error presentations must also be taken into consideration.

Most studies have so far focused on the results from the writers – the final text. However, in the paper we argue that grammar checkers are very well worth studying with methods focusing on users in process.

### 7.2.2 Methods

This paper presents three different methods used for evaluating tools like the grammar checker Granska. The first evaluation is based on an analysis of Granska's alarms on texts, which by the author or the media are assigned a genre, in these cases sport news, international news, public authority texts, popular science and student essays. This evaluation corpus comprised 201 019 words. The texts were manually scrutinized and this result was compared with Granska's alarms using the measurements recall and precision.

The second study is a so-called formative study of two grammar checkers: Granska and the Swedish grammar checker in Microsoft Word. The aim was to use the study as a part in the on-going design process of Granska. The study focused on users' responses to false alarms, wrong diagnoses and multiple suggestions from the programs.

The last study focused on cognitive revision processes in computer-aided editing. This study is mainly qualitative and focuses on how well users' revision processes are supported by a language tool like Granska. Think-aloud protocols were used to track revision processes. The difference between revision when using a grammar checker and revision without a grammar checker was studied.

### 7.2.3 Findings and conclusions

In the textual evaluation, we found that the overall recall on the five genres was 52% and the precision was 53%. This result can be compared with the evaluation on the Swedish grammar checker in Microsoft Word. However, this grammar checker was only evaluated on news papers texts, and it showed higher precision (70%) but much lower recall (35%).

An error type like split compounds is detected with quite low precision in most genres by Granska, except in the student essays. This is due to the fact that student essays actually contain split compounds, which is not always the case in the other genres.

Results from the second study suggest that several conflicting diagnoses and correction proposals seem to be a limited problem for the users if one of the proposals is correct. False alarms from the programs seem to be of variable difficulty for the users. False alarms signaling spelling errors are easier to deal with than more complicated error types.

In the third study, subjects made further changes in the text when using the grammar checker. It helped them in defining and diagnosing problems that they had problems in diagnosing manually. They also corrected more problems that they failed to correct without the grammar checker. When subjects choose not to change, it was often on style problems.

No single evaluation method gives an exhaustive answer to all important research questions. The studies made in this paper were all conducted inside the laboratory. Considering the social nature of revision and writing future studies must consider methods that study writing in more realistic settings outside the laboratory.

## 7.3 Paper 3: Automatic evaluation of robustness and degradation in tagging and parsing

### 7.3.1 Aims and background

The aim of the study was to evaluate how general text analyzers like part-of-speech taggers and surface syntactical parsers deal with ungrammatical sentences. In addition, the study tried to bring some light on how different text analyzers degrade when more and more errors are introduced in the text.

In the study the TnT PoS-tagger, the Brill PoS-tagger and a partial and shallow parser for Swedish called Granska Text Analyzer (GTA) were used. Part-of-speech taggers are used in many applications, and many applications have to deal with authentic language, which to some extent contains different kinds of errors.

### 7.3.2 Methods

In the absence of a Swedish treebank we annotated about 15 000 words from SUC. We used the output from GTA as a staring point for annotation of basic phrase types and clause boundaries. The analysis of GTA is surface-oriented and identifies several types of phrases in Swedish: adverbial phrases, adjective phrases, infinitive verb phrase, noun phrases, prepositional phrases, and limited verb phrases. The internal structure of phrases is parsed when appropriate. In addition, clause boundaries are identified. This output was manually corrected.

To prepare the experiments, we were not only missing a treebank, a corpus with annotated errors would also have been a good resource for this kind of evaluation. To mitigate this, we used artificial errors to build up an error corpus. A program called Misplel was used to introduce artificial errors. Spelling errors were chosen, as they are belonging to an error type, which is existent in many written languages.

Furthermore, automatic spelling correction is not a reliable means to correct errors in authentic texts. Hence, general text analyzers must deal with spelling errors. Artificial spelling errors will also always result in an invalid word, while another error type might introduce a not necessarily ungrammatical sentence, but one with a different interpretation. Even though the error results in an ungrammatical sentence, the surrounding words might form a different interpretation.

In the experiments, we also used a baseline tagger and a baseline parser. The part-of-speech annotations from the SUC texts were also used as a kind of tagger. Different combinations of the taggers and the parser were used on texts with different levels of errors introduced (0%, 1%, 2%, 5%, 10% and 20%), in an n-iteration test.

### 7.3.3 Findings and conclusions

In the evaluations, the robustness of the different taggers and the parser have been shown. They are all quite robust with regard to sentences where spelling errors occur. They degrade gracefully, with no dramatic drop in performance. In the taggers, unknown words are analyzed with statistics of frequent suffixes, which seem to be a key factor when handling spelling errors. If the spelling error is introduced in the first or middle part of a word, the taggers can still analyze it properly. Experiments also showed that the parser's performance relied heavily on the performance of the taggers.

## 7.4 Paper 4: Grammar checking for Swedish second language learners

### 7.4.1 Aims and background

The aim of this study was to present and then compare three different methods for grammar checking on a Swedish learner corpus. Initial studies on smaller corpus samples with the grammar checker Granska showed that the main problem is low recall. The conclusion drawn was that is was not possible to change Granska to meet the requirement of higher recall. Hundreds of hours of work have been spent on fine-tuning the rules of Granska, and increasing recall by rebuilding Granska was out of the question. Initial evaluations of Granska showed that several error types are unpredictable and very hard to detect with rules, especially in second language writers' texts. Instead, two new statistical methods were developed; ProbGranska (Bigert & Knutsson, 2002) and SnålGranska (Sjöbergh & Knutsson, 2005).

### 7.4.2 Methods

Granska is based on handcrafted rules for error detection. Granska consists of a spelling checker, a statistical part-of-speech tagger, and about 350 manually constructed rules. Granska must be considered as a state-of-art grammar checker, with fast and accurate algorithms for detection, diagnosing and correction of several frequent error types in Swedish texts.

ProbGranska looks for errors by comparing the scrutinized text with known correct texts from the SUC corpus. ProbGranska detects improbable sequences of part-of-speech tags, using frequencies collected from SUC. However, suffering from data sparseness, this procedure produced too many false alarms. To mitigate this problem phrase transformations were developed in the Granska rule language. When a rare trigram occurs, possible phrase transformations are applied in order to eliminate the rare trigram, which quite often occurs as result of more complex phrases. Longer phrases like "this lovely fast machine" is transformed to "the machine". In addition, the clause boundary detector of GTA is used to eliminate the problem with rare trigrams crossing clause boundaries. These two actions limit the complexity of the sentence, and the amount of possible tag trigrams.

SnålGranska is based on a different approach, which treats grammar checking as a tagging problem. Artificial errors are introduced in the data set used for training. Again SUC is used, but this time twice; first as a corpus of correct language, and second as a corpus containing a lot of errors. The combinations of these two are used for the training of SnålGranska. The correct corpus is necessary in order to avoid overtraining on errors. The FnTBL classifier (Ngai & Florian, 2001) was used in our experiments as an error tagger. The tagging task is to disambiguate words that have two possible interpretations; as correct or as erroneous. Simple rules for generating the artificial errors are used, in this case split compounds and agreement errors are introduced in the corpus. This method for building a grammar checker can be used for any language that has a corpus, which is annotated with part-of-speech tags, and where the errors can be generated with simple rules.

### 7.4.3 Findings and conclusions

In the evaluation of the three different methods 10 000 words from the SSM-corpus part of the CrossCheck corpus were used. The Swedish grammar checker in MS Word 2000 was also used as a kind of baseline. The results show that the different grammar checkers detect different errors.

One finding is that an ensemble grammar checker would be very useful, if higher recall is required. High recall can be achieved by relying on only one detection from one of the grammar checkers as an indication of an error.

A bottleneck for all the methods used is that the same kind of treatment of the part-of-speech tagging problem is used. How to tag ungrammatical

sentences seems to be an important problem to focus on in future research on grammar checking.

## 7.5 Paper 5: Designing and developing a language environment for second language writers

### 7.5.1 Aims and background

The aim of the study was to get an understanding on how language technology can be used in computer assisted language-learning environments. The grammar checker Granska was used as an example on language technology, with the objective to aid second language writers in their free text production. In addition, the design of a new learning environment based on the findings from the field study was an important objective.

In the study the computer is viewed as a tool, and two of the questions addressed were how the participants in study use available language tools and how these tools mediate the learner's understanding of the new language. One focus in the study was the participants' errors; the errors are a rich source when one wants to get an understanding of the learner's current perspective of the target language. Using a grammar checker can help the participants to notice the gap between their interlanguage and the target language. Awareness of errors is also one part of the pedagogical concept called focus on form (Long & Robinson, 1998). Focus on form means to draw the learner's attention to grammatical form while conducting meaningful tasks, i.e. communicative tasks.

### 7.5.2 Methods

The participants of the study were instructed to use Granska whenever they wanted in their course assignments. Granska was used with a web interface, which enabled the participants to use their ordinary word processors, of course with the cost of a quite static interface to the grammar checker.

The method for data collection was based on questionnaires, interviews, text collection, and a judgment procedure. The judgment procedure was conducted by the participants on the alarms from the grammar checker on their own writing. The participants task was to judge every alarm from the grammar checker, and also the three parts of its feedback; detection, diagnosis, and correction proposals. The participants' texts were collected in two versions; a version written before the session with Granska, and a version after the session with Granska. This procedure made it possible to

investigate to what extent the participants followed the advice from Granska, and how this was related to the judgments they made on Granska's alarms. In addition to the judgments, which were on a scale from 1 to 5; the participants made 150 comments on Granska's alarms in total. The data collection was based on the participants' willingness to take part in the study. This resulted in different levels of commitment to the given instructions.

### 7.5.3 Findings and conclusions

Observing what happens when a false alarm occurs is complex. The participants have followed Granska's advice on some false alarms, while other false alarms have been clearly rejected. Unfortunately, false alarms indicating word order errors seem to be extra hard to judge for the participants. However, the true alarms from Granska were in majority, and the participants seem to be satisfied with them. An interesting observation is that active participants gave higher grades than participants that had used Granska only a few times. Another important result is that the grammar checker is not the only tool the participants need; other language tools might also be useful in free text production.

Based on the results from the study, design principles for a language environment for second language writers were identified. These design principles were used in the development of the prototype Grim. Grim was designed for focus on form using the both grammar checking and visualization of grammatical categories. Another design principle was to provide the user with different tools to gain linguistic information, for instance lexical information and examples of language use through concordances.

# 8 Discussion and conclusions

The grammar checker Granska is central in this thesis work. Although its method for error detection is not unique, it has several features that must be considered as strong contributions to the field. The work with Granska has an important perspective of language engineering, where robustness and efficiency of the linguistic algorithms are central in the design and development processes.

Granska has been invaluable in the different evaluations conducted in this thesis. The possibility to change, improve and develop Granska has driven much research both on the integration of language tools in writing and learning environments and on the development of methods for grammar checking.

## 8.1 To write rules or not to write rules

The work on the linguistic rules in Granska must be seen as an investment for realistic evaluations with a robust language tool, and in the development of new language tools such as many of the tools in Grim. As discussed elsewhere in this thesis, to construct the general error pattern is not the hard part in the development of error rules; it is the fine-tuning of the rules that matters. It is a delicate work, with intricate linguistic considerations and grammar engineering. The struggle between recall and precision is one of the main problems in this work.

While the rules of Granska are the result of much iteration on evaluation material like error collections and proofread text collections, the parser GTA is a stand-alone application that is more or less an application coming out for free from Granska. My conclusion is that it is much easier to write rules when the main task of the rules is not to determine if the language fragment at hand is grammatical or not.

Constructing rules for different tasks in natural language processing has for long been a dominant approach to tackle NLP-problems. However, machine learning has become an important competitor during the last ten years. In the line of this evolution several questions arise. To start with,

which method is the most cost-effective? Is it cheaper to annotate a corpus, and then train a machine learner on it, or is it more effective to construct rules manually in a powerful rule language?

There exist some studies trying to answer this question (Brill & Ngai, 1999; Ngai & Yarowsky, 2000). Ngai and Yarowsky found that annotation and machine learning is more efficient and more successful than rule writing. Brill and Ngai's conclusions are that unskilled human rule writers can quickly write rules that perform quite close to learned rules. The problem is that humans have problems in formulating rules that identify less frequent constructions. Both studies are based on computer science students (not linguists) and the linguistic phenomena to annotate or formulate rules for were noun phrase chunks. Studies with linguists and more complex tasks, like error detection, might give different answers.

From my own experience, I draw the conclusion that annotations are easier to discuss with other researchers, while rules are much more personal, and harder for others to understand. My opinion is also that some problems are suitable to use rules for while other are hard to capture with rules, and statistical and machine learning methods seem to be an attractive alternative in many cases (cf. Manning 2002). Writing rules for the basic phrases in a sentence or for checking the agreement within the noun phrase seem to be a suitable work for the grammarian, while writing rules for a problem like split compounds can be more questioned. The context and the meaning are much more important when searching for split compounds than for noun phrase agreement or verb chain errors, which I consider as much more isolated errors than split compounds, word order errors or predicative disagreement.

This is also clear in the evaluations that have been carried out with Granska. The evaluations showed much better results on noun phrase disagreement and verb chain errors than on the other error types. If the part-of-speech tagger identifies the correct sequence of word classes, the error detection rules will find most errors on these error types. This also uncovers the fact that the scope of the general analyzers (i.e. part-of-speech taggers and parsers) is to a great extent the scope of the error detection rules. In order to find more complex error types with error detection rules, the grammaticality must be controlled with a broader view not only based on the analysis of forms, but also including more of context and meaning, the two factors that according to Manning (2002) also have to be considered when judging grammaticality.

The comparison of Granska, ProbGranska and SnålGranska in Paper 4 showed that the quality of handcrafted rules on some error types cannot be ignored. The precision of Granska's rules is higher than the other two methods. On the other hand, the three methods complement each other, and an ensemble of different error detection methods is promising. This together with better general methods for handling grammaticality might be the way to improve grammar checking systems in future.

## 8.2 Language tools in a language environment

Granska is central even in the language environment Grim. It was the studies with Granska that influenced much of the design of Grim. However, it was also the lack of adequate feedback and misleading feedback from Granska that made us start thinking of other tools that could support writing and learning.

In the field study in Paper 5, we saw that it was problematic for the participants that Granska gave feedback on some texts, and nearly none on other texts. For the participants, the fragmentary feedback contributed to a general lack of trust in the program's language possibilities. To meet this question of trust, the integration of other tools might help as well as improvements of the grammar checking algorithms. A tool that analyzes word classes and phrases will nearly always give the user some feedback. From the study, it was apparent that it was quite natural for the second language users to use different sources of linguistic information. They used dictionaries, asked the teacher and friends, and consulted books when writing their texts. Adding different kinds of language tools, and integrating them in an interactive language environment is the foundation of Grim.

Grim is not pedagogically neutral. The tools that are important in Grim are included in the system with the purpose to support different forms of learning. However, the learners can use them as they like, Grim is not controlling how, when and if the user is using them. When teachers want the learners to use Grim, they must invent their own instructions and pedagogical settings for the usage of the program.

Second language acquisition/learning and pedagogy is a field with no straight answers to questions related to the most effective way to teach and study a second language. Evaluation of learning effects of a specific pedagogy, or even a computer program, is a difficult task; the variables and factors are many. A current learning activity cannot be isolated from other activities that are not necessarily focusing on learning, but have learning effects. Even though not all teachers embrace language-learning

environments, the learners will probably use them, if they are available. And who will stop the learners from using tools that they believe will help them when using the second language?

## 8.3 Future work

One of the leading researchers in computer assisted language learning and second language writing is Mark Warschauer. His famous last words in (Matsuda et al., 2003) is "Researchers investigating technology and second language learning writing will have plenty to keep them busy". I have to agree with him. The advent of computers and communicational networks like Internet have increased the complexity of second language writing, but also increased the possibilities for both writers and researchers. Writers have now many new opportunities with for instance tools like the ones presented in this thesis, and also many new ways to communicate with other second language learners and native speakers. However, technology is still immature, and studies of language tools in both individual and collaborative activities are important in order to get an understanding of their mediating role in language learning as well as improving current learning environments.

### 8.3.1 The need for annotated error corpora

Researchers can use new technology in many ways both for data collection and to analyze data in the field of second language learning (Grant & Ginther, 2000). In this interface between corpus linguistics and language technology many new possibilities for research open up (Borin & Prütz, 2004). However, for a language like Swedish there is an urgent need for annotated resources, and for the development of language tools in particular there is a need for corpora where errors are identified and classified. Error analysis is a complicated task, and every tag set for error annotation can be criticized, but some critique can be avoided be trying to collect naturalistic data and error descriptions rather than error diagnoses. Dagneaux et al (1998) argue that computer-aided error analysis is the way forward in order to produce better grammar checkers:

"Before one can hope to produce 'L2 aware' grammar and style checkers, one needs to have access to comprehensive catalogues of authentic learner errors and their respective frequencies in terms of types and tokens" (Dagneaux et al, 1998, p. 165).

A first step to language tools that are more "aware" of Swedish as a second language (L2) is the collection of material of second language writers' texts compiled in the CrossCheck corpus. However, a lot of work remains to be done in annotating the errors in this corpus.

It is hard to see that we will get enough annotated errors in a reasonable time, and an attractive alternative is therefore to use artificial errors to develop better language tools. In addition, to use an ensemble of all available spelling and grammar checkers developed for Swedish might be the best starting point for a Swedish annotated error corpus, including texts from both native and second language writers.

### 8.3.2 Two perspectives on language

In this thesis, two perspectives on language are employed. The perspective used when designing the study on using language tools in second language writing and the resulting design of a learning environment (Paper 5) is based on focus on form for second language learning and a more general perspective on writing and learning based on sociocultural theory. Focus on form as well as sociocultural theory emphasize a view of language as a social activity, and that language is best learned through communication, in tasks that make sense for humans. However, when it comes to developing the algorithms for the language tools, this is an unrealistic perspective, and the dominating perspective in this thesis is a view of language as an object, in one sense only as a collection of data that are processed by linguistic algorithms. Language tools cannot participate in a social activity, they cannot negotiate and they do not understand written communication. They have to focus on the product and its surface. One important objective for the future is to develop a design of language tools that communicates this two-fold view of language.

## 8.4 Concluding remarks

The work in this thesis leads to the following conclusions:

- A hybrid grammar checker like Granska based on a general statistical language analysis and handcrafted error detection rules stands strong against both commercial and methodological alternatives.
- Language tools seem to be useful in a second language writing. The tools seem to fit into the pedagogical method called focus on form.
- An ensemble of methods seems to be a strong alternative in future language tools, both as combined algorithms and as competing tools in a language environment like Grim.
- Different ways of evaluating language tools are fruitful, but still many studies have to be carried out to get an understanding of the mediation of language tools in language learning. The design of

learning environments based on language technology has to be improved through new evaluations.

- The issue of grammaticality is crucial both for general language analyzers, methods for error detection and for the users of language tools. The question of how to support the user's judgments of grammaticality has to be seriously addressed in the design and development of writing and learning environments.
- False alarms and missed errors are still essential problems to deal with, and the need for annotated Swedish error corpora as a base for the development of better methods for error detection, diagnosis and correction is urgent.

# References


Ahrenberg, L. (1990). *A grammar combining phrase structure and field structure.* In the proceedings of the 13th International Conference on Computational Linguistics (COLING 1990), Helsinki, Finland.

Almqvist, I., & Sågvall Hein, A. (1996). *Defining ScaniaSwedish - a controlled language for truck maintenance.* In the proceedings of the 1st International Workshop on Controlled Language Applications, CLAW 96, KU Leuven, Belgium.

Arppe, A. (2000). *Developing a grammar checker for Swedish.* In the proceedings of the 12th Nordic Conference in Computational Linguistics, Trondheim, Norway.

Atwell, E. S. (1987). *How to detect grammatical errors in a text without parsing it.* In the proceedings of the 3rd EACL, Copenhagen, Denmark.

Bigert, J. (2005). *Automatic and unsupervised methods in natural language processing.* Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden.

Bigert, J., & Knutsson, O. (2002). *Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge.* In the proceedings of the RObust Methods in Analysis of Natural language Data, ROMAND 2002, Frascati, Italy.

Birn, J. (1998). Swedish Constraint Grammar. A short presentation. Retrieved December 12th, 2000, from http://www.lingsoft.fi/doc/swecg/intro/

Birn, J. (2000). *Detecting grammar errors with Lingsoft's Swedish grammar checker.* In the proceedings of the 12th Nordic Conference in Compuational Linguistics, Trondheim, Norway.

Bondi Johannessen, J., Hagen, K., & Lane, P. (2002). *The performance of a grammar checker with deviant language input.* In the proceedings of the 19th International Conference on Computational Linguistics (COLING), Taipei, Taiwan.

Borin, L. (2002). *What have you done for me lately? The fickle alignment of NLP and CALL.* In the proceedings of the EuroCALL 2002 pre-conference workshop on NLP in CALL, Jyväskylä, Finland.

Borin, L., & Prütz, K. (2004). New wine in old skins? A corpus investigation of L1 syntactic transfer in learner language. In G. Aston, S. Bernardini & D. Stewart (Eds.), *Corpora and language learners* (pp. 67-87). Amsterdam: John Benjamins.

Bourdon, M., Sylva, L. D., Gagnon, M., Kharrat, A., Knoll, S., & Maclachlan, A. (1998). *A case study in implementing dependency-based grammars.* In the proceedings of the Workshop on the Processing of Dependency-based Grammars, COLING-ACL'98, Montreal, Canada.

Brants, T. (2000). *TnT - a statistical part-of-speech tagger.* In the proceedings of the 6th Applied NLP Conference, ANLP-2000, Seattle, USA.

Bredenkampf, A., Crysmann, B., & Petrea, M. (2000). *Looking for errors: A declarative formalism for resource-adaptive language checking.* In the proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000), Athens, Greece.

Brill, E. (1995). Transformation-based error-eriven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics, 21*(4), 543-565.

Brill, E., & Ngai, G. (1999). *Man [and woman] vs. machine: A case study in base noun phrase learning.* In the proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, Maryland, USA.

Brock, M. N. (1993). Three disk-based text analyzers and the ESL writer. *Journal of Second Language Writing, 2*(1).

Brodda, B. (1983). *An experiment with heuristic parsing of Swedish.* In the proceedings of the First Conference of the European Chapter of the Association for Computatlonal Linguistics (EACL), Pisa, Italy.

Bustamente, F. R., & León, F. S. (1996). *GramCheck: A grammar and style checker.* In the proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, Denmark.

Carlberger, J., Domeij, R., Kann, V., & Knutsson, O. (submitted). The Development and Performance of a Grammar Checker for Swedish: A Language Engineering Perspective.

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

Carlson, A. A., Rosen, J., & Roth, D. (2001). *Scaling up context-sensitive text correction.* In the proceedings of the Innovative Applications of Artificial Intelligence Conference, Seattle, WA, USA.

Chapelle, C. A. (2001). *Computer Applications in Second Language Acquisition. Foundations for teaching, tesing and research.* Cambridge, Great Britain: Cambridge University Press.

Chodorow, M., & Leacock, C. (2000). *An unsupervised method for detecting grammatical errors.* In the proceedings of the NAACL'01, Seattle, USA.
Chomsky, N. (1957). *Syntactic structures*. The Hague: Mouton & Co., Publishers.
Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA, USA: The M.I.T. Press.
Dagneaux, E., Denness, S., & Granger, S. (1998). Computer-aided error analysis. *System, 26*, 163-174.
de Smedt, K., & Rosén, V. (2000). *Automatic proofreading for Norwegian: The challenges of lexical and grammatical variation.* In the proceedings of the 12th Nordic Conference in Compuational Linguistics, Trondheim, Norway.
DiCamilla, F. J., & Lantolf, J. P. (1994). The linguistic analysis of private writing. *Language Sciences, 16*(3/4), 347-369.
Domeij, R. (2003). *Datorstödd språkgranskning under skrivprocessen: Svensk språkkontroll ur användarperspektiv.* Ph.D. thesis, Stockholm University, Stockholm, Sweden.
Domeij, R., Knutsson, O., Carlberger, J., & Kann, V. (2000). *Granska - an efficient hybrid system for Swedish grammar checking.* In the proceedings of the 12th Nordic Conference on Computational Linguistics, Trondheim, Norway.
Domeij, R., Knutsson, O., & Öhrman, L. (1999). *Inkongruens och felaktigt särskrivna sammansättningar - en beskrivning av två feltyper och möjligheten att detektera felen automatiskt.* In the proceedings of the Svenskans beskrivning 24, Linköping, Sweden.
Doughty, C., & Williams, J. (1998). Pedagogical choices in focus on form. In C. Doughty & J. Williams (Eds.), *Focus on Form in Classroom Second Language Acquisition* (pp. 197-261). New York, USA: Cambridge University Press.
Ejerhed, E. (1999). Finite state segmentation of discourse into clauses. In A. Kornai (Ed.), *Extended Finite State Models of Language*. Cambridge: Cambridge University Press.
Ejerhed, E., Källgren, G., Wennstedt, O., & Åström, M. (1992). *The linguistic annotation system of the Stockholm-Umeå Corpus project.* Umeå, Sweden: Department of General Linguistics, University of Umeå.
Ellis, R. (1997). *Second Language Acquisition*. Oxford, Great Britain: Oxford University Press.
Flower, L. S., & Hayes, J. R. (1981). A cognitive process theory of writing. *College Composition and Communication, 37*(1), 365-387.

Gambäck, B. (1997). *Processing Swedish sentences: A unification-based grammar and some applications.* Ph.D. thesis, Royal Institute of Technology, Stockholm.

Gass, S. M. (1994). The reliabilty in L2 grammaticality judgments. In E. E. Tarone, S. M. Gass & A. D. Cohen (Eds.), *Research Methodology in SLA* (pp. 303-322). Hillsdale, NJ, USA: Lawrence Erlbaum.

Gellerstam, M., Cederholm, Y., & Rasmark, T. (2000). *The bank of Swedish.* In the proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000), Athens, Greece.

Grant, L., & Ginther, A. (2000). Using computer-tagged linguistic features to describe L2 writing differences. *Journal of Second Language Writing, 9*(2), 123-145.

Grefenstette, G., & Tapanainen, P. (1994). *What is a word, What is a sentence? Problems of tokenization.* In the proceedings of the The 3rd International Conference on Computational Lexicography, Budapest, Hungary.

Haas, C. (1996). *Writing technology. Studies on the materiality of literacy*. New Jersey, USA: Lawrence Erlbaum Associates.

Haas, C. (1999). On the Relationship Between Old and New Technologies. *Computers and Composition, 16*(2), 209-228.

Hammerton, J., Osborne, M., Armstrong, S., & Daelemans, W. (2002). Introduction to special issue on machine learning approaches to shallow parsing. *Journal of Machine Learning Research, Special Issue on Shallow Parsing, 2*, 551-558.

Han, N., Chodorow, M., & Leacock, C. (2004). *Detetcting errors in English article usage with a maximum entropy classifier trained on a large diverse corpus.* In the proceedings of the 4th International Conference on Language Resources and Evalauation (LREC 2004), Lisbon, Portugal.

Hardt, D. (2001). *Transformation-based learning of Danish grammar correction.* In the proceedings of the Recent Advances of Natural Language Processing 2001 (RANLP-2001), Tzigov Chark, Bulgaria.

Hassel, M. (2001). *Internet as corpus: Automatic contruction of a Swedish news corpus.* In the proceedings of the 13th Nordic Conference on Computational Linguistics (Nodalida'01), Uppsala, Sweden.

Heidorn, G. E. (2000). Intelligent writing assistance. In R. Dale, H. Moisl & H. Somers (Eds.), *Handbook of Natural Language Processing* (pp. 181-207). New York, NY, USA: Marcel Dekker.

Huang, J. H., & Powers, D. (2001). *Large scale experiments on correction of confused words.* In the proceedings of the 24th Australasian Conference on Computer Science, Gold Coast, Queensland, Australia.

Hyland, K. (2003). Genre-based pedagogies: A social response to process. *Journal of Second Language Writing, 12*(1), 17-29.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., & Isahara, H. (2003). *Automatic error detection in the Japanese learners' English spoken data.* In the proceedings of the ACL-03 Interactive Posters and Demonstrations, Sapporo, Japan.

James, C. (1998). *Errors in Language learning and Use: Exploring Error Analysis*. London, Great Britain: Longman.

Jensen, K., Heidorn, G. E., Miller, L. A., & Ravin, Y. (1983). Parse fitting and prose fixing: Getting hold on ill-formedness. *American Journal of Computational Linguistics, 9*(3-4), 123-136.

Jones, M. P., & Martin, J. H. (1997). *Contextual spelling correction using latent semantic analysis.* In the proceedings of the 5th Conference on Applied Natural Language Processing, Washington, DC, USA.

Kann, V., Domeij, R., Hollman, J., & Tillenius, M. (2001). Implementation aspects and applications of a spelling correction algorithm. *Quantitative Linguistics, 60*, 108-123.

Karlgren, J. (2000). *Stylistic experiments for information retrieval.* Ph.D. thesis, Stockholm University, Stockholm, Sweden.

Karlsson, F. (1990). *Constraint Grammar as a framework for parsing running text.* In the proceedings of the 13th International Conference on Computational Linguistics, COLING-90, Helsinki, Finland.

Karlström, P., & Cerratto Pargman, T. (forthcoming). *Mediating role of language tools in learners' dialogic interaction*. Unpublished manuscript.

Knutsson, O. (2001). *Automatisk språkgranskning av svensk text.* Ph.Lic. thesis, Royal Institute of Technology, Stockholm.

Knutsson, O. (2002). *Inkongruens i predikativ - både rätt och fel.* In the proceedings of the Svenskans beskrivning 25, Åbo, Finland.

Kokkinakis, D., & Johansson-Kokkinakis, S. (1999). *A cascaded finite-state parser for syntactic analysis of Swedish.* In the proceedings of the 9th European Chapter of the Association of Computational Linguistics (EACL), Bergen, Norway.

Kollberg, P. (1998). *S-notation - a computer-based method for studying and representing text composition*. Ph.Lic. thesis, Royal Institute of Technology, Stockholm, Sweden.

Koskenniemi, K. (1983). *Two-level morphology: A general computational model for word-form recognition and production.* Ph.D. thesis, University of Helsinki, Helsinki.

Krashen, S. (1982). *Principles and practices in second language acquisition*. Englewood Cliffs, NJ, USA: Prentice Hall.

Krashen, S. (1994). The input hypothesis and its rivals. In N. Ellis (Ed.), *Implicit and explicit learning of languages*. London, Great Britain: Academic Press Limited.

Källgren, G. (1979). *Innehåll i text.* Ph.D. thesis, Stockholm University, Stockholm, Sweden.

Källgren, G. (1992). *Making maximal use of morphology in large-scale parsing: the MorP parser*. Stockholm, Sweden: Department of Linguistics, Stockholm University.

Lantolf, J. P. (2000). Introducing sociocultural theory. In J. P. Lantolf (Ed.), *Sociocultural Theory and Second Language Learning* (pp. 1-26). Oxford, Great Britain: Oxford University Press.

Lantolf, J. P., DiCamilla, F. J., & Ahmed, M. (1997). The cognitive function of linguistic performance: Tense/Aspect use by L1 and L2 speakers. *Language Sciences, 19*, 153-165.

Levy, M. (1997). *Computer-assisted Language Learning: Context and Conceptualization*. Oxford, Great Britain: Oxford University press.

Lexin. (2003). Lexin - ett lexikon för invandrarundervisning. Retrieved 2005-09-13, from http://www-lexikon.nada.kth.se/

Lindberg, I. (2001). Några forskningsperspektiv på interaktionens roll i andraspråksinlärning. In U. B. Uhlmann, B. Söderman & Å. Vikström (Eds.), *P.S. Postscriptum. Språkliga studier till minnet av Elsie Wiijk-Andersson*. Uppsala, Sweden: Hallgren & Fallgren.

Lindberg, I., & Skeppstedt, I. (2000). Ju mer vi lär tillsammans - rekonstruktion av text i smågrupper. In H. Åhl (Ed.), *Svenskan i tiden - verklighet och visioner*. Stockholm, Sweden: HLS Förlag.

Lindberg, J., & Eriksson, G. (2004). *CrossCheck-korpusen - en elektronisk svensk inlärarkorpus*. In the proceedings of the ASLA Conference 2004, Södertörns högskola, Sweden.

Long, M. H. (1981). Input, interaction, and second language acquisition. In H. Winitz (Ed.), *Native language and foreign language acquisition* (Vol. 279, pp. 259-278): Annals of the New York Academy of Sciences.

Long, M. H., & Robinson, P. (1998). Focus on form: Theory, research, and practice. In C. Doughty & J. Williams (Eds.), *Focus on Form in Classroom Second Language Acquisition* (pp. 15-41). New York, USA: Cambridge University Press.

Mangu, L., & Brill, E. (1997). *Automatic rule acquisition for spelling correction.* In the proceedings of the 14th International Conference on Machine Learning, Nashville, TN, USA.

Manning, C. (2002). Probabilistic syntax. In R. Bod, J. Hay & S. Jannedy (Eds.), *Probabilistic Linguistics*. Cambridge, MA, USA: MIT Press.

Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press.

Matsuda, P. K., Canagarajah, A. S., Harklau, L., Hyland, K., & Warschauer, M. (2003). Changing currents in second language writing research: A colloquium. *Journal of Second Language Writing, 12*(2), 151-179.

Megyesi, B. (2002a). *Data-driven syntactic analysis: methods and applications for Swedish.* Ph.D. thesis, Royal Institute of Technology, Stockholm.

Megyesi, B. (2002b). Shallow parsing with PoS taggers and linguistic features. *Journal of Machine Learning Research, Special Issue on Shallow Parsing, 2*, 639-668.

Moro, Y. (1999). The expanded dialogic sphere: Writing activity and authoring of self in Japanese classrooms. In Y. Engeström, R. Miettinen & R.-L. Punamäki (Eds.), *Perspectives on Activity Theory* (pp. 165-182). Cambridge, United Kingdom: Cambridge University Press.

Nerbonne, J. (2002). Computer-Assisted Language Learning and Natural Language Processing. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*: Oxford University Press.

Ngai, G., & Florian, R. (2001). *Transformation-based learning in the fast lane.* In the proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001), Pittsburgh, USA.

Ngai, G., & Yarowsky, D. (2000). *Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking.* In the proceedings of the 38th Annual Meeting of the Associations for Computational Linguistics (ACL'2000), Hong Kong.

Nivre, J. (2000). Sparse data and smoothing in statistical part-of-speech tagging. *Journal of Quantitative Lingustics, 7*(1), 1-17.

Nivre, J., Hall, J., & Nilsson, J. (2004). *Memory-based dependency parsing.* In the proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004), Boston, USA.

Normark, M. (2002). *Using technology for real-time coordination of work; A study of work and artifact use in the everyday activities of SOS Alarm*. Ph. Lic. thesis, Royal Institute of Technology, Stockholm.

Nunn, B. (2001). Task-based methodology and sociocultural theory. *The Language Teacher Online* Retrieved 2005-07-28, from http://ltsc.ph-karlsruhe.de/Nunn.pdf

Olson, D. R. (1994). *The world on paper: The conceptual and cognitive implications of writing and reading*. Cambridge, USA.

Olson, D. R. (1995). Writing and the mind. In J. V. Wertsch, P. d. Río & A. Alvarez (Eds.), *Sociocultural studies of mind* (pp. 95-123). New York, USA: Cambridge University Press.

Paggio, P. (2000). *Spelling and grammar correction for Danish in SCARRIE.* In the proceedings of the 6th Conference on Applied natural language processing, Seattle, Washington, USA.

Park, J. C., Palmer, M., & Washburn, G. (1997). *An English grammar checker as a writing aid for students of English as a second language.* In the proceedings of the 5th Conference on Applied Natural Language Processing, Washington, DC, USA.

Pihl, E., Rastas, H., & Rockberg Tjernberg, A. (2003). *Betydelsen av feedback i Grim - en interaktiv lärmiljö med fokus på det svenska språket*. Unpublished manuscript.

Ravin, Y. (1993). Grammar errors and style weakness in a text-critiquing system. In K. Jensen, G. E. Heidorn & S. D. Richardsson (Eds.), *Natural Language Processing: The PLNLP Approach* (pp. 14-27). Norwell, MA, USA: Kluwer Academic Publishers.

Reid, J., Lindstrom, P., McCaffrey, M., & Larson, D. (1983). Computer-assisted text-analysis for ESL students. *CALICO Journal, 1*(3), 40-42, 46.

Richardson, S., & Braden-Harder, L. (1993). The Experience of Developing a Large-Scale Natural Language Processing System: Critique. In K. Jensen, G. E. Heidorn & S. D. Richardsson (Eds.), *Natural Language Processing: The PLNLP Approach* (pp. 77-89). Norwell, MA, USA: Kluwer Academic Publishers.

SAOL. (1986). *Svenska Akademiens ordlista*. Stockholm, Sweden: Nordstedts förlag.

Schneider, D. A., & McCoy, K. F. (1998). *Recognizing syntactic errors in the writing of second language learners.* In the proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL-98), Montréal, Canada.

Selinker, L. (1972). Interlanguage. *International Review of Applied Linguistics*(10), 209-231.

Severinson Eklundh, K. (1986). *Dialogue processes in computer mediated communication. A study of letters in the Com system.* Ph. D. thesis, Linköping University, Linköping, Sweden.

Sjöbergh, J. (2003). *Combining POS-taggers for improved accuracy on Swedish text.* In the proceedings of the 14th Nordic Conference on Computational Linguistics, Reykjavik, Iceland.

Sjöbergh, J., & Knutsson, O. (2005). *Faking errors to avoid making errors: very weakly supervised learning for error detection in writing.* In the proceedings of the Recent Advances in Natural Language Processing 2005, RANLP 2005, Borovets, Bulgaria.

Skeppstedt, I. (2005). *Kan deltagare som studerar svenska för invandare (sfi) och svenska som andraspråk (sas) inom den grundläggande vuxenutbildningen använda och få hjälp och stöd av Grim?* Unpublished manuscript.

Smagorinsky, P. (1994). Think-aloud protocol analysis: Beyond the black box. In P. Smagorinsky (Ed.), *Speaking About Writing: Reflections on Research Methodology.* (pp. 3-19). Thounsand Oaks, California, USA: SAGE Publications, Inc.

Sofkova Hashemi, S. (2003). *Automatic Detection of Grammar Errors in Primary School Children's Texts.* Ph.D. thesis, Göteborg University, Gothenburg.

Sofkova Hashemi, S., Cooper, R., & Andersson, R. (2003). *Positive Grammar Checking: A Finite State Approach.* In the proceedings of the CICLing 2003, Mexico City, Mexico.

Sparc Jones, K. (2001). Automatic language and information processing: rethinking evaluation. *Natural Language Engineering, 7*(1), 29-46.

Swain, M. (1998). Focus on form through conscious reflection. In C. Doughty & J. Williams (Eds.), *Focus on Form in Classroom Second Language Acquisition* (pp. 64-81). New York, USA: Cambridge University Press.

Swain, M. (2000). The output hypothesis and beyond: Mediating acquisition through collaborative dialogue. In J. P. Lantolf (Ed.), *Sociocultural Theory and Second Language Learning* (pp. 97-114). Oxford, Great Britain: Oxford University Press.

Säljö, R. (2000). *Lärande i praktiken: ett sociokulturellt perspektiv* (first ed.). Stockholm: Bokförlaget Prisma.

Sågvall Hein, A. (1981). *An overview of the Uppsala chart parser version 1 (UCP-1).* Uppsala, Sweden: Department of Linguistics, University of Uppsala.

Sågvall Hein, A. (1998). *A chart-based framework for grammar checking. Initial studies.* In the proceedings of the 11th Nordic Conference in Computational Linguistics, Copnehagen, Denmark.

Sågvall Hein, A. (1999). *A grammar checking module for Swedish.* Uppsala, Sweden: Department of linguistics.

Sågvall Hein, A., Almqvist, A., Forsbom, E., Tiedemann, J., Weijnitz, P., Olsson, L., et al. (2002). *Scaling up an mt prototype for industrial*

*use. Databases and data flow.* In the proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas, Spain.

Tschichold, C., Bodmer, F., Cornu, E., Grosjean, F., Grosjean, L., Kubler, N., et al. (1997). *Developing a new grammar checker for English as a second language.* In the proceedings of the ACL'97 Workshop: From Research to Commercial Applications: Making NLP Work in Practice, Madrid, Spain.

van Lier, L. (2000). From input to affordance: Social-interactive learning from an ecological perspective. In J. P. Lantolf (Ed.), *Sociocultrual Theory and Second Language Learning*. Oxford, Great Britain: Oxford University Press.

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

Voutilainen, A. (2001). *Parsing Swedish.* In the proceedings of the 13th Nordic Conference on Computational Linguistics (Nodalida'01), Uppsala, Sweden.

Wertsch, J. V. (1991). *Voices of the mind. A sociocultural approach to mediated action*. Cambridge, MA, USA: Harvard University Press.

Wertsch, J. V. (1998). *Mind as action*. New York, USA: Oxford University Press.

Åström, M. (1996). A probabilistic tagger for Swedish using the SUC tagset. In H. Feldweg & E. Hinrichs (Eds.), *Lexikon und Text* (pp. 245-256). Tübingen: Max Niemeyer Verlag.

# Paper 1

# *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/`

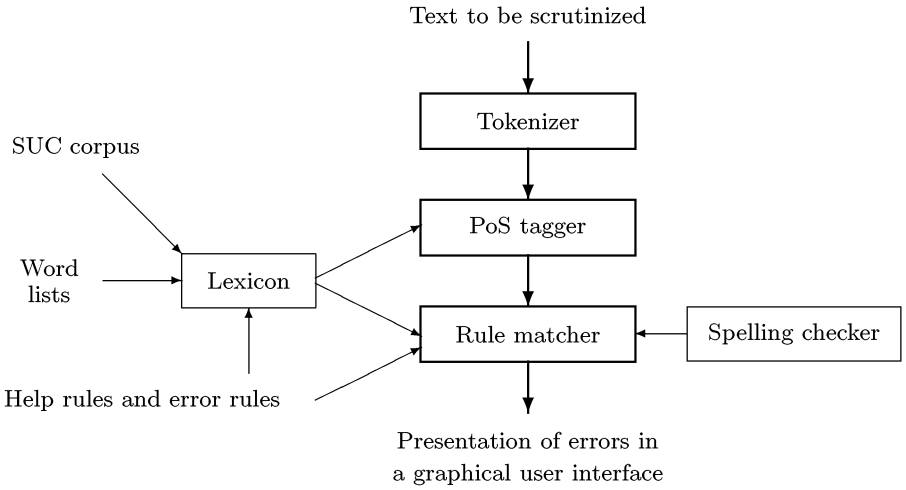


Fig. 1. An overview of the GRANSKA system.

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

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 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.

# Paper 2

# Different Ways of Evaluating a Swedish Grammar Checker

**Rickard Domeij, Ola Knutsson and Kerstin Severinson Eklundh**

Department of Numerical Analysis and Computer Science
Royal Institute of Technology
SE- 100 44 Stockholm, Sweden
{domeij, knutsson, kse}@nada.kth.se

**Abstract**

Three different ways of evaluating a Swedish grammar checker are presented and discussed in this article. The first evaluation concerns measuring the program's detection capacity on five text genres. The measures (precision and recall) are often used in evaluating grammar checkers. However, in order to test and improve the usability of grammar checking software, they need to be complemented with user-oriented methods. Consequently, the second and the third evaluations presented in the article both involve users. The second evaluation focuses on user reactions to grammar error presentations, especially with regard to false alarms and erroneous error identification. The third and last evaluation focuses on problems in supporting users' cognitive revision processes. It also examines user motives behind choosing to correct or not to correct problems highlighted by the program. Advantages and disadvantages of the different evaluation methods are discussed.

## 1. Introduction

Tools for checking mechanics, grammar and style in writing are widely used as an integrated part of common word processors. Until recently, advanced tools have been lacking for smaller languages, such as Swedish. However, there are now one commercial grammar checker, Grammatifix (Arppe, 2000), and two research prototypes available, Scarrie (Sågvall-Hein, 1998) and Granska (Domeij et al, 2000).

There are many reasons for further research and development of authoring aids. First, the need for such aid has increased, especially when the computer as a writing tool has reached many new and different user groups, for example high school students and second language learners. Secondly, before adapting the grammar checkers to new user groups, there is a need for more sophisticated methods for evaluating the functionality and usability of the programs and their effects on users' ability and practices of revision in writing.

This paper will focus on evaluations made in relation to the development of the Swedish grammar checker Granska. We argue that the evaluation of grammar and style checking must go further than merely measuring the functionality by measures of precision and recall, and thus seriously address the issue of usability. By giving examples of three different studies made during the development of Granska, the advantages of using a broader approach to evaluation are demonstrated.

## 2. The evaluated system

Granska is a grammar checker for Swedish developed at the Royal Institute of Technology in Sweden. It is together with other language tools integrated in a writing environment supporting different aspects of the writing process. Granska combines probabilistic and rule-based methods to achieve high efficiency and robustness (see also Carlberger & Kann, 1999). Using special error rules, the system can detect a number of Swedish grammar problems and suggest corrections for them that are presented to the user together with instructional information.

The interface of a grammar checker serves several important functions. On a general level, it gives a picture of the program's capabilities and way of working for the user. More specifically, it communicates with the user about the errors encountered, describing these errors as well as giving suggestions for correcting them.

Importantly, the interface is also where the program communicates with the user's writing process. If properly designed, it provides for a transparent and easy switch between the grammar checking and other processes of text composition. Although it constitutes a part of the general process of revision, there is no predefined place in writing to which grammar checking can be confined. This is because writing is a highly complex, recursive and individual activity (Flower & Hayes, 1981). Accordingly, the interface should provide means for invoking the grammar checker interactively at any time, and for going back to writing without delay or inconvenience. We have considered these aspects of the design of the interface in our work on the Granska system.

Granska is presently being adapted for second language learners of Swedish. The evaluations presented in the article have been made during different stages in the development of Granska. The development is still an ongoing process, involving recurrent evaluation of functionality and usability.

## 3. Related research

In other research areas such as information retrieval and information extraction, evaluation methods have been seriously developed in relation to forums such as TREC, MUC and, for Europe, CLEF. Notably, the grammar checking area is short of empirical evaluative efforts of this kind, although some efforts have been made (see the Eagles report for an overview of different evaluations and evaluation methods).

Earlier studies of grammar and style checking software have involved measuring the program's error detection capacity in terms of precision (i.e. error detection correctness) and recall (i.e. error coverage) (see e.g. Kukich, 1992; Birn, 2000; Richardson & Braden-Harder, 1993). The need of measuring the quality of correction alternatives and instructions has also been recognized (see

e.g. Kohut & Gorman, 1995; TEMAA-report, 1997 pp. 34).

Richardson & Braden-Harder (1993) take different text genres into account and report large differences in error detection rates between for instance texts from professional writers and freshman compositions. They also report that professionals are more forgiving to wrong proposals than students.

Kohut & Gorman (1995) evaluate the effectiveness of several commercial grammar and style packages in the writing of business students. In this study, real errors detected by the program were further classified as correctly identified (incorrect usage accurately classified by the program) or incorrectly identified (incorrect usage misclassified by the program). For the correctly identified errors, the remedial advice was rated by experts as very helpful, helpful or not helpful.

Other studies have investigated the impact of specific software on the quality of produced text (see Kohut & Gorman, 1995 for an overview). The studies have often been conducted in pedagogical settings, comparing improvements in text quality between two groups of students, one group using a grammar checker, the other not. Some studies report positive effects while others report no measurable effects at all. The mixed results may be due to problems in controlling the relevant variables or not using sufficiently sensitive variables.

An advantage with the measurements of recall and precision mentioned above is that they are well defined. On the other hand, the results are hard to interpret. Do users prefer high precision before high recall, or perhaps the other way around? The truth is that we do not know what users prefer before we study them. Therefore, measures of precision and recall can only be a starting point. On top of that, aspects such as user abilities and needs, variability of text genres and user groups, the complexity of error types and error presentations must also be taken into consideration.

Although most of the studies mentioned above in some sense are user-oriented in their approach, none of the studies did study real users during computer-aided revision. To get a deeper understanding of user related issues in grammar checking, we decided to study users in process.

## 4. Three evaluations

In the following three sections, we will present three different evaluations performed in different stages during the development of the Swedish grammar checker Granska. The first evaluation concerns precision and recall of error rules on five text genres for the Swedish grammar checker Granska. It focuses on the functionality of the system and aims at measuring its error detection capacity for three error types across different genres. This study was made during the error rule implementation phase of the project.

The second and the third evaluations involve users in two different ways. The second evaluation is formative and focuses on user reactions to error presentations, especially with regard to false alarms and erroneous error identification. It relies on observational methods complemented with tape recordings of users thinking aloud. The evaluation was performed during the work with error presentations and correction alternatives.

The third and last evaluation focuses on problems in supporting users' cognitive revision processes. The main research question addressed here is if a grammar and style checker has the capacity to support the user in managing three important steps in the revision process: detection, diagnosis and correction. It also examines user motives behind choosing to correct or not to correct problems highlighted by the program. Revision processes and motives for revising are studied by analyzing think-aloud protocols in depth. This study was carried out early in the design process using an experimental prototype of the grammar checker. The work with coding and analyzing the vast amount of data went on during later phases. The study both served to inform and evaluate design decisions.

After the three evaluations have been presented in closer detail in the following sections, the different methods used will be further discussed.

## 5. Evaluation 1: A text analysis evaluation

Granska was evaluated on five text genres comprising about 200 000 words (Knutsson, 2001). The detections and diagnoses from Granska on these texts were manually examined. The result indicates differences in the outcome of the grammar checking between text genres. In the following text, recall is defined as 'detected errors/all errors' and precision is defined as 'correct alarms/all alarms'.

Collecting and annotating an evaluation corpus are a demanding task, and one problem is to obtain texts that are under revision. The texts in the material have to varying extent been proofread, which is demonstrated in the evaluation results on the different text genres. The text genres were sport news, international news, public authority text, popular science text and student essays. The evaluation corpus contained 418 syntactic errors.

The largest groups of error types in the evaluation material are the following: disagreement within the noun phrase (17%), split compounds (18%), verb chain errors (21%), missing words (13%) and so called context-sensitive spelling errors (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 material. We are continuously working on expanding the error coverage of Granska, and presently focusing on errors specific for second language learners.

The overall recall for all errors in the five genres is 52% and the precision is 53%. The results from the most frequent error types are presented in table 1.

| Error type | Sport news | International news | Public authority | Popular science | Student essays | All texts |
|---|---|---|---|---|---|---|
| Verb chain errors | 100/91 | 100/71 | 75/86 | 100/78 | 100/76 | 97/83 |
| Split compounds | 100/11 | -/0 | 71/42 | 60/27 | 40/67 | 46/39 |
| Disagreement within NPs | 88/38 | 100/11 | 100/25 | 100/37 | 74/72 | 83/44 |

Table 1. Recall/precision percentages on five text genres for three frequent error types in the material.

There is a big difference between the results from the different text genres. Granska achieves the best results on verb chain errors (e.g. *Han har spela fiol/He has play violin*). Verb chain errors got a recall ranging from 75% in public authority texts to 100% in sport news. This may indicate that these errors are easier to find and correct than for instance split compounds (e.g. *Jag samlar bok märken/I'm collecting book marks*).

The results on split compounds need further explanations. Split compounds are very difficult to detect without generating false alarms, and therefore there needs to be quite a few errors in the texts in order to achieve a precision over 50%. Student texts contain more errors than the other texts, which results in a precision of 67% and a recall of 40%. Looking at the same error type in public authority texts gives a precision of 42% and a recall of 71%. Moreover, in international news, Granska only generated false alarms and no detections, which can be explained by the fact that there were no split compounds occurring at all in international news text.

Comparing the results with other evaluations is difficult because of factors such as different languages, text types, the complexity of error types, error frequencies in the texts and more. However, some comparisons might be interesting despite all difficulties. The Critique system for English has also been evaluated (Richardson & Braden-Harder, 1993) on different text genres with lower accuracy on texts from professional writing (about 40%) and much higher on freshman composition (72%). The results from the evaluation of Critique are in line with Granska's results on different text genres. For Swedish, an evaluation made by Birn (2000) has been conducted on newspaper texts, and reports a recall of 35% and a precision of 70%. The system evaluated was the Swedish grammar checker in Microsoft Word. The precision is higher than Granska's overall results, while recall is lower, which may suggest different design choices made during the program development in the intricate trade-off between recall and precision. One notable difference is that Word's grammar checker does not address the complex error type split compounds, which Granska does with some loss of precision as a result.

## 6. Evaluation 2: A formative study of two grammar checkers

During the development of Granska a formative evaluation was carried out. The evaluation consisted of a small user study involving Granska and a commercial grammar checker (Knutsson, 2001). Five users participated in the study. The users were all experienced writers and had all, to some extent, used grammar checking tools before.

Direct observation was used complemented with tape recordings of users thinking aloud. The tape recordings were used as background information in the study, which focuses on the observations. The user's task was to use the two grammar checkers for checking a text containing errors possible for at least one of the programs to detect. When an alarm from the grammar checker occurred, the users could either accept or reject the alarm. They could also correct the errors themselves if they found it suitable.

The study focused on users' responses to false alarms, wrong diagnoses and multiple suggestions from the programs. These three problems are important to study during the development process of a grammar checker. They all address the problem of the trade-off between recall and precision.

If false alarms really are a problem for the users, we have to increase precision, which also means decreased recall, because of the inverse relation between the two measures. If users found multiple diagnosis and suggestions problematic we have to implement a decision mechanism that presents only one diagnosis and suggestion, with the risk of presenting one erroneous diagnosis and suggestion instead of two or more possible error interpretations. In other words, should the user or the program select among alternative interpretations?

One rather common example of multiple diagnoses and suggestions are split compounds versus disagreement within NPs. Consider for example the sentence *Jag vill ha många vy kort* (eng. *I want many post cards*). It could be interpreted as a split compound *vy kort* (*post card*) or as a number disagreement between *många* (*many*) and *vy* (*post*). In the study, the commercial grammar checker did not present multiple diagnoses but Granska did in form of a list of alternatives presented to the user. At this stage in the development of Granska, we were seeking a metric that could rank and possibly avoid alternative interpretations of an error. Before implementing such a metric, we wanted to know how users reacted to multiple interpretations.

Results suggest that several conflicting diagnoses and proposals seem to be a limited problem for the users if one of the proposals is correct. It only took the users' a minimal amount of extra time to select the correct alternative among several. This gave us valuable information for the further development of Granska. Since there seemed to be limited need for implementing a metric for choosing only one diagnosis and suggestion, our further efforts in the development process were

concentrated on improving the program with regard to false alarms and missed.

Moreover, the results showed that some users seem to need only the detection from a grammar checker, and are able to make the correction in the text by themselves. Surprisingly often, they corrected the text according to the programs' proposals, but instead of inserting them by pressing the buttons in the interface, they typed the correction directly into the text.

False alarms from the programs seem to be of variable difficulty for the users. Easily judged false alarms from the spell checker did not cause users to change the text, but false alarms on more complicated error types sometimes fooled users to change and follow the advice from the two grammar checkers.

## 7. Evaluation 3: A study of cognitive revision processes in computer-aided editing

In the third evaluation, we wanted to take a closer look at the cognitive processes behind the observed revision behavior. The study is mainly qualitative and focuses on how well human revision processes are supported by writers' aids from a cognitive perspective. Think-aloud methodology is used to track revision processes (such as detection, diagnosis and correction) during computer aided editing. An analysis of the think-aloud protocols reveals how well a grammar checker manages to support these processes; when and why the tool succeeds or fails to support the writer in revising highlighted problems in the text.

The research is influenced by the work of Hayes et al. (1987) in which a detailed psychological model of the revision process is presented and used in studying revision. The revision process is described as being composed of the following three subprocesses: task definition, evaluation and strategy selection. Three stages in the process are pinpointed as problematic, especially for inexperienced writers, i.e. detecting, diagnosing and revising problems in text. In Hill et al (1991) the same theoretical framework and methodology is used to study on-line editing.

The aim of the present study was to examine the usefulness and effect of writers' aids more closely in the light of this framework. It was a further development of a previous study using a similar design but without think-aloud methodology (Domeij, 1998).

In the present study, 11 university students with considerable experience in writing were asked to revise a letter, first using pen and paper, then using computer aids. The letter was originally a negative response from the authorities to a young girl who had asked for permission to marry before the age of sixteen. For the study, the letter had been prepared to contain 37 problems in mechanics, grammar and style, all of which could be analyzed by the computer tool.

Think-aloud methodology was used to track the revision process both during manual and computer-aided editing. The design made it possible to compare the number of changes that subjects made to planted problems with and without computer aid. Most importantly, it made it possible to find explanations to the observed revision behavior by analyzing the think-aloud protocols. Thus, the study combined quantitative and qualitative methods.

The quantitative results showed that, on average, subjects changed 85% of all problems when using the grammar checker, compared to 60% without it. Subjects refrained from changing 15% of all problems although urged to attend to them by the grammar checker. Why did subjects sometimes change further problems when using the grammar checker, and sometimes not? Some interesting answers were found by analyzing the think-aloud protocols.

Subjects made further changes when using the grammar checker because it aided them in a) detecting problems they had missed in the manual revision, b) defining and diagnosing problems that they had problems diagnosing manually, c) correcting problems that they had failed to find corrections for manually, and d) detect, diagnose and correct problems which they did not know before. Negative effects were also observed, as when subjects were fooled to change because of a false alarm. The results also suggest that changes can be less extensive and more surface-oriented when using the grammar checker.

There were two reasons why subjects did sometimes not change when using the grammar checker: a) the reviser wanted to change but failed because of insufficient instructional support from the grammar checker, or because of other kinds of interactional problems such as pressing the wrong button, b) the reviser chose not to change because he or she did not find the response correct or useful in the present situation. The second situation was by far the most commonly observed.

When subjects choose not to change, it was most often in response to problems in style, where some could be seen to disagree heatedly to the advice from the computer. For example, when one of the writers got the suggestion from the program to consider changing "ingå äktenskap" (eng. "enter into marriage") to "gifta sig" (eng. marry) in order to avoid an excessively bureaucratic style, he responded: "No, I don't agree to that because this is kind of a legal text!"

Interestingly, though, the influence of the tool on the number of changes made in style varied greatly between different subjects. While some writers made almost no changes in style, even though they were urged to attend to them by the computer tool, other writers changed many problems in style such as "enter into marriage" both with and without computer support.

Data from the think-aloud protocols suggest that these differences are related to how different writers define the task of revising. Those who made many changes in style were observed to be more reader-oriented than those who refrained from changing. Clearly, writers showed conflicting views about which style is appropriate in a letter from the authorities: a traditional style characterized by high formality and intransparancy, or a less formal reader-oriented style characterized by clarity. This inhomogeneous nature of style even within genres, make style checking problematic.

## 8. Discussion and future work

It is our hope that the three evaluative studies presented have convincingly shown the advantages of studying users and combining different qualitative and quantitative methods in the evaluation of authoring aids. While the first study contributed to evaluating the

functionality of the error detection capacity, the two other evaluations informed us of how users reacted to different detected problems and their presentations.

The results of the two later studies are interesting mainly in two respects: 1) they use process-tracking methods that shed light on the cognitive processes involved in computer-aided revision, and 2) they pinpoint interactional problems that must be addressed and attended to in designing more useful grammar checking systems. Thus they enabled us to make important design choices based on user data, as what rules to include in the program, what error presentations and instructions to improve on, or how to present different correction alternatives to the user.

The third study was indeed time consuming in its detailed analysis of think-aloud data, but it also produced interesting and general results concerning problems in supporting different cognitive processes in revision. For example, the problem of supporting different users' task definitions involving style decisions was seen to be very complex because of writers' conflicting views of which style to use within the genre. Although style checking is an interesting problem, it needs further research along this line before it can be effectively supported by computers.

We will pursue the cognitive perspective further in the near future as we are adapting the program to writers with Swedish as a second language. Similar studies as those presented here will be performed on users from this group.

Undoubtedly, there are methodological problems associated with using think-aloud data. There is, for example, reason to be careful when generalizing observations made using think-aloud methodology because of the unnatural situation forced upon writers as they are made to speak out their thoughts during the act or writing. However, think-aloud methodology still remains the most effective way of generating data of the thinking processes involved in revision (cf. Hayes & Flower 1983).

When carrying out an evaluation of a grammar checking system, it is very difficult from a methodological perspective to recreate the conditions of an individual writer, using the system as the need arises. Therefore our evaluations have instead been carried out in a partly simulated mode, where writers get a draft text to analyze and correct. This means that some challenging issues of evaluation have not yet been dealt with.

Revision is not necessarily a one-man show. We must not let the cognitive perspective make us forget the socially embedded nature of writing, as the before-mentioned problems in supporting style checking remind us of. In practice, revision, and more generally writing, is performed in a specific social situation, e.g. in a newspaper office or a second language class. Most often, it also involves negotiation and cooperation between people who may contribute to the task in different ways, as for example in newspaper editing and peer reviewing where someone writes a text and others take part in reviewing and revising it.

When designing a grammar checker as an integrated tool in a system for supporting writing, the context and the cooperative practices of revision should be taken into consideration. Evidently, the editor at the newspaper and his colleagues need other support and aid for their work processes, than the second-language student and his peers in their class at high school. Therefore, in future evaluations we are also considering using ethnographical methods of studying work practices in realistic settings.

In developing and evaluating authoring aids there is need for multidisciplinary approaches using several complementary research methods (see Monk & Gilbert 1995 and Smagorinsky 1994 for an overview of theoretical perspectives and research methods used within human computer interaction and writing research respectively). No single evaluation method gives an exhaustive answer to all important research questions. In this paper we have presented three different ways of evaluating a Swedish grammar checker. In doing that we hope to have contributed somewhat to a broader understanding of the problems involved in evaluating authoring aids.

## 9. References


Arppe, A., 2000. Developing a grammar checker for Swedish. In *Proc. 12th Nordic Conference in Computational Linguistics, Nodalida-99*. Trondheim, pp. 13-27.

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

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

Domeij, R., 1998. Detecting, diagnosing and correcting low-level problems when editing with and without computer aids. In *TEXT Technology*, vol 8, no. 1, pp. 14-25. Wright State University, Celina, USA.

Domeij, R., Knutsson, O., Carlberger, J. & Kann, V., 2000. Granska – an efficient hybrid system for Swedish grammar checking. I *Proc. 12th Nordic Conference in Computational Linguistics, Nodalida-99*. Trondheim, pp. 49–56.

EAGLES Evaluation of Natural Language Processing Systems report 1996. http://www.ilc.pi.cnr.it/EAGLES96/browse.html. Last updated 090696.

Flower, L. S., & Hayes, J. R., 1981. A cognitive process theory of writing. *College Composition and Communication*, 32, pp. 365-387.

Hayes, J. R., Flower, L., Schriver, K., Stratman, J. & Carey, L., 1987. Cognitive processes in revision. In: S. Rosenberg (Ed.), *Advances in applied psycholinguistics: Vol. 2*. pp. 176–240. New York: Cambridge University Press.

Hayes, J. R. & Flower, L., 1983. Uncovering Cognitive Processes in Writing: An Introduction to Protocol Analysis. In Mosenthal, Tamer & Walmsley (Eds.), *Research on Writing: Principles and Methods*. New York: Longman.

Hill, C. A., Wallace, D. L., and Haas, C., 1991. Revising on-line: Computer technologies and the revising process. Computers and Composition, 9(1), pp. 83-109.

Knutsson, O., 2001. Automatisk språkgranskning av svensk text. Licentiate thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm.

Kohut, G. & Gorman, K., 1995. The effectiveness of leading grammar/style software in analysing business students' writing. *JTBC,* pp. 341-361. July 1995.

Kukich, K., 1992. Techniques for automatically correcting words in text. *ACM Computing surveys,* Vol. 24, No. 4, pp. 377–439.

Monk, A.F. & Gilbert, N., 1995 *Perspectives on HCI: Diverse Approaches*. London: Academic Press.

Richardson, S & Braden-Harder, L., 1993. The Experience of Developing a Large-Scale Natural Language Processing System: Critique. In Jensen, K. Heidorn, G. E. Richardson, S. D. (eds.), *Natural Language Processing: The PLNLP Approach*, pp. 77-89.

Smagorinsky, P. (Ed.)., 1994. *Speaking about Writing: Reflections on Research Methodology*. Thousand Oaks, California: Sage Publications.

Sågvall Hein, A., 1998. A chart-based framework for grammar checking. Initial Studies. I *Proc. 11th Nordic Conference in Computational Linguistics, Nodalida-9*8, Copenhagen, pp.68–80.

TEMAA -A Testbed Study of Evaluation Methodologies: Authoring Aids. Final report. 1997. http://cst.dk/projects/temaa/D16/d16exp.html. Last updated: October 1997.

# Paper 3

# Automatic Evaluation of Robustness and Degradation in Tagging and Parsing

**Johnny Bigert, Ola Knutsson, Jonas Sjöbergh**
Department of Numerical Analysis and Computer Science
Royal Institute of Technology, Sweden
{johnny,knutsson,jsh}@nada.kth.se

Keywords: automatic evaluation, robustness, spelling errors, tagging, shallow parsing

## Abstract

We address the topic of automatic evaluation of robustness and performance degradation in parsing systems. We focus on one aspect of robustness, namely ill-formed sentences and the impact of spelling errors on the different components of a parsing system. We propose an automated framework to evaluate robustness, where ill-formed and noisy data is introduced using an automatic tool and fed to the parsing system. With increasing levels of noise, the performance of a system will inevitably degrade, and the question is to what extent? The experiments show a graceful degradation in performance for both state-of-the-art taggers used and a Swedish shallow parser. The automated nature of the evaluation allows easy and reproducible evaluation of the individual components of a parsing system.

## 1 Introduction

Comparable and reproducible evaluations and experiments are important foundations for research. Valid and comparable evaluation of NLP-systems is often hard to achieve without a great portion of manual work. Many languages lack resources for evaluation and even for languages with available material, new types of evaluations based on existing material need a lot of manual labor. Furthermore, manual interference with the evaluation material may in fact decrease the validity of the evaluation.

In this paper we present an automatic evaluation method focusing on the robustness of parsers for syntactic analysis. The robustness of a parser is defined here as robustness against ill-formed input, which is only one aspect as pointed out by Menzel 95. The proposed evaluation technique is fully automatic, with no need for manual annotation or inspection. It relies on a general tool, which introduces different kinds of errors into a text. The errors can be spelling or grammatical errors, but we have focused on spelling errors resulting in non-existing words to avoid some ambiguity problems, as explained later.

To demonstrate the evaluation method, we applied it on a shallow parser for Swedish. The experiments are presented as a glass box evaluation, where the performance of the over-all system is presented as well as the performance of the components, such as part-of-speech taggers. All tests are conducted with various levels of errors introduced, under which the system performance degradation is measured. Since this paper focuses on evaluation methodology, we do not address how the introduced errors affect syntactic structure. Nevertheless, automatic evaluation of the effects on syntactic structure is indeed an interesting topic for future work.

The main contribution of this paper is an automatic method for the evaluation of robustness and degradation in tagging and parsing. The evaluation method is useful for determining the performance impact from individual components and thus, is suitable for e.g. on-going development of parsing systems. We argue that introducing noise in text with automatic means is a valid and reliable technique for evaluating a system's robustness.

This paper consists of five parts. To begin with, we discuss evaluation methods for parsing and subsequently, the proposed method for automatic evaluation is presented. In the third part, the shallow parser used in the experiments is briefly outlined. In the fourth section, experimental methodology and the experiments are described. Lastly, we present the results.

### 1.1 Parser Evaluation

Parsers have been evaluated in the parser community using different methods over the years, from simple test suites to treebanks (for a comprehensive overview see Carroll *et al.* 98). A prerequisite for the evaluation method presented here is an annotated corpus, preferably a treebank. For Swedish, there exist small forests annotated with functional structure (Nivre 02). Unfortu-

nately, none of these were suitable for our experiments, which are based on an annotation scheme for constituency. We adopted the IOB tag set (Ramshaw & Marcus 95) for row-based phrase assignment of Swedish constituents.

Using different text genres for testing is one way of evaluating robustness (see e.g. Li & Roth 01). The problem with such an evaluation is the lack of control of the differences between the genres; is sports harder to parse than economical news? Another way is to compare well-written and proofread texts with more noisy texts. One problem with such a comparison is the difficulty in comparing the two text types without manual analysis of the errors. In the following section we will present a method that compares proofread text and text with different degrees of automatically introduced noise. The method is one way of evaluating a specific kind of robustness of an NLP-system.

## 2 The Proposed Method

When processing unrestricted natural language data, rare and malformed constructions and spelling errors occur quite frequently. We want to assess the degree to which a parser can handle these difficulties, and to which degree the analysis fails for the context surrounding a deficiency. To this end, we require a source of texts with annotated language errors. One problem is that annotated resources containing errors do not exist for all languages. Furthermore, the texts must also be annotated with parse information to be able to serve as a comparison to the output of a parser. Where data of sufficient size exists, it is often not sufficiently annotated in terms of parse trees. Another problem with annotated errors is the difficulty in determining, for a given error, what the corrected text should be. Normally, we do not know what the intention of the writer was, which introduces further ambiguity in the parse trees.

### 2.1 Introducing Errors

To avoid the problems mentioned, we decided to start from correct text and from there, introduce errors using an automated tool, MISSPLEL (Bigert *et al.* 03).

MISSPLEL is a general-purpose error introducer, able to introduce most error types produced by humans, such as spelling errors resulting in existing and non-existing words, with or without a change in part-of-speech (PoS) tag, competence errors such as sound-alike errors and context sensitive errors such as feature agreement errors and word order errors.

Most of these error types have the effect of altering the original semantics of the sentence, especially when a word is misspelled, resulting in an existing word. This is indeed a problem since the parse tree of the new sentence may differ from that of the original sentence. Thus, there is a possibility that the output of the parse system is in fact correct even though it differs from the annotated parse tree. We approach this problem by restricting the introduced errors to spelling errors that result in non-existing words only. Hence, the new sentence does not have a straightforward interpretation. Nevertheless, the most plausible interpretation of the new sentence is that of the original text.

During introduction of errors, every word containing alphanumeric characters has an equal chance of being misspelled. Given a word to misspell, we choose a position in the word randomly. To simulate performance errors caused by keyboard mistypes, we choose between insertion, deletion and substitution of a single letter as well as transposition of two letters (Damerau 64). When substituting a letter for another, letters close on the keyboard are more often mistaken than those far apart. The situation is similar for insertion, since this is often caused by pressing two keys at the same time. Thus, keys closer on the keyboard have a higher confusion probability when substituting and inserting. For further details, see Bigert *et al.* 03.

The automatic introduction of spelling errors permits us to choose the percentage of errors in a text. It also allows us to choose any text or text type provided that it is annotated with parse trees. Furthermore, it allows for an $n$-iteration test to determine how different phrase types degrade with increasing number of errors, and thus, minimizing the influence of chance.

### 2.2 Evaluation

To gather the necessary data, we used an automated evaluation tool called AUTOEVAL (Bigert *et al.* 03), which is a general purpose evaluation

```
Viktigaste (the most important) APB|NPB  CLB
redskapen (tools)               NPI      CLI
vid (in)                        PPB      CLI
ympning (grafting)              NPB|PPI  CLI
är (is)                         VCB      CLI
annars (normally)               ADVPB    CLI
papper (paper)                  NPB|NPB  CLI
och (and)                       NPI      CLI
penna (pen)                     NPB|NPI  CLI
,                               O        CLB
menade (meant)                  VCB      CLI
han (he)                        NPB      CLI
.                               O        CLI
```

Figure 1: *Example sentence showing the IOB format.*

```
((CL (NP (AP Viktigaste) redskapen)
     (PP vid (NP ympning))
     (VC är)
     (ADVP annars)
     (NP (NP papper) och (NP penna)))
 (CL  ,
     (VC menade)
     (NP han)) . )
```

Figure 2: *The text from Figure 1 in a corresponding bracketing format.*

tool for structured data (e.g. row-based data or XML) configurable using a script language.

The output from the GTA parser was given in the so-called IOB format (Ramshaw & Marcus 95). See Figure 1 and 2 for a sentence with phrase labels and clause boundaries in the IOB and bracketing format, respectively. As an example, `NPB|PPI` means that the beginning (`B`) of an noun phrase (`NP`) is inside (`I`) a prepositional phrase (`PP`). Thus, the rightmost phrase is the topmost node in the corresponding parse tree. The phrase types are explained in Section 3.

Using AutoEval, we gathered information on tag accuracy, full row parse accuracy, clause boundary identification accuracy as well as precision, recall and F-scores for all phrase types.

The statistics for individual phrase types were calculated as follows. Given a phrase type to evaluate, all other phrases were removed. The same was done for the correct, annotated parse, and the results were then compared. The parser was successful if and only if they were identical. For example, we are looking at phrases of type NP. If the correct parse is `APB|NPB|NPI` (an adjective phrase in a noun phrase inside another noun phrase), the parse `NPB|APB|NPI` would be correct since the adjective phrase is ignored, while the parse `APB|NPI|NPI` would be incorrect since the leftmost NP differs.

Since many parsers rely heavily on the performance of a part-of-speech tagger, we include several taggers with different behavior and characteristics. Apart from taggers representing state-of-the-art in part-of-speech tagging, we also include a perfect tagger and a baseline tagger. The perfect tagger does nothing more than adopt the original tags found in the annotated resource. The baseline tagger is constructed to incorporate little or no linguistic knowledge and was included to establish the difficulty of the tagging task.

Parsing different texts may result in different accuracy for the parser at hand. To provide a clue to the inherent difficulty of a text, we require a baseline for the parsing task. The perfect tagger, the baseline tagger and the baseline parser are further discussed in the experiments section.

In the experiments below, we use six error levels (0%, 1%, 2%, 5%, 10%, 20%). For a given error level $p$, we introduce spelling errors (resulting in non-existing words only) in a fraction $p$ of the words. This procedure is repeated $n = 10$ times to mitigate the influence of chance and to determine the standard deviation of the accuracy and F-scores.

With increasing amounts of erroneous text, the performance of the parser will degrade. In order to be robust against ill-formed and noisy input, we want the accuracy to degrade gracefully with the percentage of errors. That is, for a parser relying heavily on PoS tag information, we aim for the parsing accuracy to degrade equal to or less than the percentage of tagging errors introduced. Of course, this is not feasible for all phrase types. For example, when the infinite marker or verb is misspelled, an infinitival verb phrase will be difficult to identify.

## 3 A Robust Shallow Parser for Swedish

Most parsers for Swedish are surface oriented, and two of them identify constituency structure. One uses machine learning (Megyesi 02) while the other is based on finite-state cascades (Kokkinakis & Johansson-Kokkinakis 99). Furthermore, two other parsers identify dependency structure using Constraint Grammar (Birn 98) and Functional Dependency Grammar (Voutilainen 01). There is also a deep parser developed in the CLE framework (Gambäck 97). We recently developed

a rule-based shallow parser called Granska Text Analyzer (GTA) (Knutsson *et al.* 03) and due to availability, GTA was used in the experiments. Unfortunately, none of the Swedish parsers could serve as a comparison, since none of them used a comparable constituency structure to that of GTA.

GTA is rule-based and relies on hand-crafted rules written in a context-free formalism. The parser selects grammar rules top-down and uses a passive chart. The rules in the grammar are applied on PoS tagged text, either from an integrated tagger or from an external source. GTA identifies constituents and assigns phrase labels. However, no full trees with a top node are built.

The analysis is surface-oriented and identifies many types of phrases in Swedish. The basic phrase types are adverbial phrases (ADVP), adjective phrases (AP), infinitival verb phrases (INFP), noun phrases (NP), prepositional phrases (PP) and verb phrases (limited) and chains (VP and VC). The internal structure of the phrases is parsed when appropriate and the heads of the phrases are also identified. PP-attachment is left out of the analysis since the parser does not include a mechanism for resolving PP-attachments. The disambiguation of phrase boundaries is primarily done within the rules, and secondly using heuristic selection and disambiguation rules.

In addition to the parsing of phrase structure, clause boundaries (CLB) are detected, resembling Ejerhed's algorithm for clause boundary detection (Ejerhed 99).

The parser was designed for robustness against ill-formed and fragmentary sentences. For example, agreement is not considered in noun phrases and predicative constructions (Swedish has a constraint on agreement in these constructions). By avoiding the constraint for agreement, the parser will not fail due to textual errors or tagging errors. Tagging errors that do not concern agreement are to some extent handled using a set of tag correction rules based on heuristics on common tagging errors.

## 4 Experiments

We used the toolbox described in Section 2 to evaluate the rule-based parser for Swedish. We used the Stockholm-Umeå corpus (SUC) (Ejerhed *et al.* 92) and chose six random texts (aa02, ac04, je01, jg03, kk03 and kk09) from three different categories, totally amounting to 15 000 words. The text categories were press articles (a), scientific journals (j) and imaginative prose (k). The part-of-speech tag set contained 149 tags. As there exists no constituency tree-bank for Swedish at present, the texts were annotated for tree structure. The texts were first run through the parser and then carefully corrected by a human annotator. The tokenization and sentence boundaries was determined by the corpus.

Since the performance of the parser depends heavily on the performance of the part-of-speech tagger, we compared tagged text from four different sources: the original corpus tags, a hidden Markov model (HMM) tagger, a transformation-based tagger and a baseline tagger. The tagger CORPUS used the original annotations in the SUC corpus, which we assume to have 100% accuracy. The HMM tagger used was TnT (Brants 00), hereafter denoted TNT. The transformation-based tagger (Brill 92) used was fnTBL (Ngai & Florian 01), denoted BRILL. The baseline tagger called BASE chose the most frequent tag for a given word and, for unknown words, the most frequent tag for open word classes. All taggers were trained on SUC data not included in the tests.

To determine the difficulty of the chosen texts, we constructed a baseline parser. To this end, we adopted the approach provided by the CoNLL chunking competition (Tjong Kim Sang & Buchholz 00), i.e. for a given part-of-speech tag, the parse chosen was the most frequent parse for that tag. Given a PoS tagged text, the data was divided into ten parts. Each part was parsed by using the original annotation for the other nine parts as training data. Furthermore, to determine the difficulty of the clause boundary identification we devised a baseline clause identifier simply by assigning `CLB` to the first word of each sentence and `CLI` to the other words.

Thus, we had four taggers (BASE, BRILL, TNT and CORPUS) and two parsers (GTA and baseline). For each combination of tagger and parser, we ran an $n$-iteration test (using $n = 10$) at each error level (0%, 1%, 2%, 5%, 10% and 20%). In each test, we extracted information about tagging accuracy, parsing accuracy, clause boundary identification and phrase identification for the individual phrase categories ADVP, AP,

| Tagger | 0% | 1% | 2% | 5% | 10% | 20% |
|---|---|---|---|---|---|---|
| BASE | 85.2 | 84.4 (0.9) | 83.5 (1.9) | 81.2 (4.6) | 77.1 (9.5) | 69.0 (19.0) |
| BRILL | 94.5 | 93.8 (0.7) | 93.0 (1.5) | 90.9 (3.8) | 87.4 (7.5) | 80.1 (15.2) |
| TNT | 95.5 | 95.0 (0.5) | 94.3 (1.2) | 92.4 (3.2) | 89.5 (6.2) | 83.3 (12.7) |

Table 1: *Accuracy in percent from the tagging task. The* CORPUS *tagger had 100% accuracy. The columns correspond to the percentage of errors introduced. Relative accuracy degradation compared to the 0% error level is given in brackets.*

| Tagger | 0% | 1% | 2% | 5% | 10% | 20% |
|---|---|---|---|---|---|---|
| BASE | 81.0 | 80.2 (0.9) | 79.1 (2.3) | 76.5 (5.5) | 72.4 (10.6) | 64.5 (20.3) |
| BRILL | 86.2 | 85.4 (0.9) | 84.5 (1.9) | 82.0 (4.8) | 78.0 (9.5) | 70.3 (18.4) |
| TNT | 88.7 | 88.0 (0.7) | 87.2 (1.6) | 85.2 (3.9) | 81.7 (7.8) | 75.1 (15.3) |

Table 2: *Accuracy in percent from the parsing task. Parsing based on the* CORPUS *tagger had 88.4% accuracy. A baseline parser using the* CORPUS *tagger had 59.0% accuracy.*

INFP, NP, PP and VC. Also, since some tokens are outside all phrases, we included an outside category (O).

## 5 Results

The most important aspect of the accuracy of the GTA parser is the performance of the underlying tagger. Most taggers were quite robust against ill-formed and noisy input as seen from Table 1. For example, at the 20% error level, TNT degraded 12.7% and BRILL degraded 15.2% relatively to their initial accuracy of 95.5% and 94.5%, respectively. The low degradation in performance is most likely due to the robust handling of unknown words in BRILL and TNT, where the suffix determines much of the morphological information. Thus, if the last letters of a word are unaffected by a spelling error, the tag is likely to remain unchanged. The robustness of the baseline tagger was not as satisfactory as it guessed the wrong tag in almost all cases (19.0% of 20%). The baseline tagging accuracy for text without errors was 85.2%.

For the parsing task, we obtained 86.2% accuracy using BRILL and 88.7% accuracy using TNT, as seen in Table 2. An interesting observation is that the accuracy of parsing using CORPUS, i.e. perfect tagging, was 88.4%, which is lower than that of TNT. The explanation is found in the way the taggers based on statistics generalize from the training data. The CORPUS tagger adopts the noise from the manual annotation of the SUC corpus, which will make the task harder for the parser. This is further substantiated below when we discuss the baseline parser.

The degradation at the 20% error level seems promising since the accuracy only dropped 15.3% using the TNT tagger. On the other hand, since the performance of TNT had already degraded 12.7% in tagging accuracy, the additional $15.3 - 12.7 = 2.6\%$ was due to the fact that the context surrounding a tagging error was erroneously parsed. This difference is the degradation of the parser in isolation. Nevertheless, the performance of the whole system is the most relevant measure, since the most accurate tagger does not necessarily provide the best input to the rest of the parsing system. As stated earlier, since this paper describes evaluation methodology only, we do not address how the errors affected the syntactic structure.

As a comparison, the baseline parser using the CORPUS tagger had 59.0% accuracy, while the TNT tagger obtained 59.2%. This further indicates that the difference between TNT and CORPUS is real and not just an idiosyncrasy of the parsing system. A system not using any knowledge at all, i.e. the baseline parser using the BASE tagger, obtained 55.5% accuracy.

As seen from Table 3, the task of clause identification (`CLB`) was more robust to ill-formed input than any other task with only 7.0% degradation using TNT at the 20% error level. This may be attributed to the fact that half the clause delimiters resided at the beginning of a sentence and thus, were unaffected by spelling errors. Of course, the

| Tagger | 0% | 1% | 2% | 5% | 10% | 20% |
|---|---|---|---|---|---|---|
| BASE | 84.2 | 84.0 (0.2) | 83.6 (0.7) | 82.9 (1.5) | 81.9 (2.7) | 79.4 (5.7) |
| BRILL | 87.3 | 87.0 (0.3) | 86.6 (0.8) | 85.6 (1.9) | 83.8 (4.0) | 80.3 (8.0) |
| TNT | 88.3 | 87.9 (0.4) | 87.5 (0.9) | 86.6 (1.9) | 85.1 (3.6) | 82.1 (7.0) |

Table 3: *F-score from the clause boundary identification task. Identification based on the* CORPUS *tagger had an F-score of 88.2%. A baseline identifier had an F-score of 69.0%. The columns correspond to the percentage of errors introduced. Relative accuracy degradation compared to the 0% error level is given in brackets.*

| Type | 0% | 1% | 2% | 5% | 10% | 20% | Count |
|---|---|---|---|---|---|---|---|
| ADVP | 81.9 | 81.3 (0.7) | 80.6 (1.5) | 78.6 (4.0) | 75.3 (8.0) | 68.4 (16.4) | 1008 |
| AP | 91.3 | 90.5 (0.8) | 89.8 (1.6) | 87.0 (4.7) | 83.1 (8.9) | 74.3 (18.6) | 1332 |
| INFP | 81.9 | 81.4 (0.6) | 80.9 (1.2) | 79.2 (3.2) | 76.0 (7.2) | 70.2 (14.2) | 512 |
| NP | 91.4 | 90.9 (0.5) | 90.2 (1.3) | 88.4 (3.2) | 85.2 (6.7) | 79.3 (13.2) | 6895 |
| O | 94.4 | 94.2 (0.2) | 93.9 (0.5) | 93.3 (1.1) | 92.1 (2.4) | 89.9 (4.7) | 2449 |
| PP | 95.3 | 94.8 (0.5) | 94.3 (1.0) | 93.0 (2.4) | 90.9 (4.6) | 85.8 (9.9) | 3886 |
| VC | 92.9 | 92.3 (0.6) | 91.5 (1.5) | 89.8 (3.3) | 86.8 (6.5) | 80.9 (12.9) | 2562 |
| Total | 88.7 | 88.0 (0.7) | 87.2 (1.6) | 85.2 (3.9) | 81.7 (7.8) | 75.1 (15.3) | |

Table 4: *F-scores for the individual phrase categories from the parse task.* TNT *was used to tag the text.*

baseline clause identifier was also unaffected by spelling errors and obtained a 69.0% F-score for all error levels. Clause identification at 0% error level achieved an 88.3% F-score (88.3% recall, 88.3% precision) using TNT.

We provide the F-scores for the individual phrase categories using TNT in Table 4. We see that adverbial (ADVP) and infinitival verb phrases (INFP) are much less accurate than others. They are also among the most sensitive to ill-formed input. In the case of INFP, this may be attributed to the fact that they are often quite long and an error introduced near or at the infinite marker or the verb is detrimental. In the count column, we provide the number of rows in which a given phrase type occurs in the annotation. For example, in the case of NP, we count the number of rows in which at least one `NPB` or `NPI` occurs.

Standard deviation was calculated for all accuracy and F-score values at each error level, by using data from the $n$ runs. Standard deviations were low for all tasks and were 0.13, 0.22 and 0.22 on the average for Tables 1, 2 and 3, respectively. The maximum standard deviation was 0.70 for the 20% error level for clause boundary identification using TNT. Standard deviation was 0.49 on the average for Table 4. The only noticeable exception was the infinitival verb phrase (INFP), which had a 2.5 standard deviation at the 20% error level using the BRILL tagger.

Note that 15 000 words may not be sufficient for a reliable conclusion on robustness. The experiments here are primarily provided to illustrate the evaluation method. The results from the evaluation are based on non-tuned output from the parser compared to the manually annotated data. Furthermore, there are still some minor inconsistencies between parser output and the annotation scheme. This is of course a source of systematic errors and will be dealt with in a near future.

## 6 Conclusions

We have described an automatic framework for testing the robustness of tagging and parsing. We introduced spelling errors resulting in non-existing words in order to feed the parsing system with ill-formed and noisy data. From this, the different components of a parsing system can be individually evaluated. With increasing levels of noise, the performance of the system will inevitably degrade. Here, we have addressed the difference between tagging the original, error-free text and tagging noisy text and found that state-of-the-art tagging is quite robust against ill-formed input. Furthermore, we have discussed the effect of tagging performance degradation on the over-all performance of parsing systems. Experi-

ments conducted on a shallow parser for Swedish exhibited graceful degradation in over-all parsing performance. To conclude, we advocate the use of the proposed automatic evaluation method to obtain fair and reliable measures of over-all parsing system performance, as well as a measure of performance of the individual components.

## References


(Bigert *et al.* 03) J. Bigert, L. Ericson, and A. Solis. Missplel and AutoEval: Two generic tools for automatic evaluation. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland, 2003.

(Birn 98) J. Birn. Swedish constraint grammar. Technical report, Lingsoft Inc, Helsinki, Finland, 1998.

(Brants 00) T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of ANLP-2000*, Seattle, USA, 2000.

(Brill 92) E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92*, pages 152–155, Trento, Italy, 1992.

(Carroll *et al.* 98) J. Carroll, T. Briscoe, and A. Sanfilippo. Parser evaluation: a survey and a new proposal. In *Proceedings of LREC 1998*, pages 447–454. Granada, Spain, 1998.

(Damerau 64) F. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

(Ejerhed 99) 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.

(Ejerhed *et al.* 92) E. Ejerhed, G. Källgren, O. Wennstedt, and M. Åström. *The Linguistic Annotation System of the Stockholm-Umeå Project.* Department of Linguistics, University of Umeå, Sweden, 1992.

(Gambäck 97) B. Gambäck. *Processing Swedish Sentences: A Unification-Based Grammar and some Applications.* Unpublished PhD thesis, The Royal Institute of Technology and Stockholm University, 1997.

(Knutsson *et al.* 03) O. Knutsson, J. Bigert, and V. Kann. A robust shallow parser for Swedish. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland, 2003.

(Kokkinakis & Johansson-Kokkinakis 99) D. Kokkinakis and S. Johansson-Kokkinakis. A cascaded finite-state parser for syntactic analysis of Swedish. In *Proceedings of the 9th EACL*, pages 245–248, Bergen, Norway, 1999. Association for Computational Linguistics.

(Li & Roth 01) X. Li and D. Roth. Exploring evidence for shallow parsing. In W. Daelemans and R. Zajac, editors, *Proceedings of CoNLL-2001*, pages 38–44, Toulouse, France, 2001.

(Megyesi 02) B. Megyesi. Shallow parsing with PoS taggers and linguistic features. *Journal of Machine Learning Research*, Special Issue on Shallow Parsing(2):639–668, 2002.

(Menzel 95) W. Menzel. Robust processing of natural language. In *Proceedings of 19th Annual German Conference on Artificial Intelligence*, pages 19–34, Berlin, Germany, 1995.

(Ngai & Florian 01) G. Ngai and R. Florian. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47, Carnegie Mellon University, Pittsburgh, USA, 2001.

(Nivre 02) J. Nivre. What kinds of trees grow in Swedish soil? a comparison of four annotation schemes for Swedish. In *Proceedings of First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 123–138, Sozopol, Bulgaria, 2002.

(Ramshaw & Marcus 95) L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey, 1995.

(Tjong Kim Sang & Buchholz 00) E. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000.

(Voutilainen 01) A. Voutilainen. Parsing Swedish. In *Proc. of Nodalida 01 - 13th Nordic Conference on Computational Linguistics*, 2001.

# Paper 4

# Grammar checking for Swedish second language learners

**Johnny Bigert  Viggo Kann  Ola Knutsson  Jonas Sjöbergh***
KTH Nada, SE-100 44 Stockholm

## Abstract

Grammar errors and context-sensitive spelling errors in texts written by second language learners are hard to detect automatically. We have used three different approaches for grammar checking: manually constructed error detection rules, statistical differences between correct and incorrect texts, and machine learning of specific error types.

The three approaches have been evaluated using a corpus of second language learner Swedish. We found that the three methods detect different errors and therefore complement each other.

## Svensk sammanfattning

Grammatikfel och kontextberoende stavfel (felstavningar som bildar riktiga ord) i texter skrivna av andraspråksinlärare är svårt att detektera automatiskt. Vi har använt tre olika angreppssätt för granskningen: manuellt konstruerade feldetekteringsregler, statistiska skillnader mellan korrekt och felaktig text, samt maskininlärning av specifika feltyper.

De tre metoderna har vi utvärderat på en korpus bestående av svenska uppsatser av andraspråksinlärare. Vi fann att metoderna upptäcker olika fel och därför kompletterar varandra väl.

## 1 Introduction

Language technology has a potential to play a major role in the process of learning a language. Until recently, the use of language technology in systems for language learning has been nearly nonexistent. However, this has not been the case with grammar checkers for second language learners

*E-mail {johnny,viggo,knutsson,jsh}@nada.kth.se
This work was done in the research project *CrossCheck – a grammar checker for second language writers of Swedish* funded by Vinnova and KTH between 2001 and 2004, see http://www.nada.kth.se/theory/projects/xcheck/

learning English (see e.g. Bolt (1992), Chen (1997), Park et al. (1997), Yazdani (1990)). The question if grammar checkers actually improve second language learners' language is still a question of debate (Chen 1997; Vernon 2000). In spite of this, we see the adaptation of grammar checking for Swedish to second language learners as a first step to put language technology in computer assisted language learning environments.

Designing a grammar checker for second language learners raises several questions. Are second language learners writing much worse (at a grammatical level) essays than for instance native writers? Are they violating the grammatical rules of Swedish in a significant different way? These questions are hard to answer without large scaled and fine-grained corpus studies. In the CrossCheck project we have developed a corpus of second language Swedish, both for studying the types of errors and testing our grammar checkers, and as a general resource for the second language learning research community (Lindberg and Eriksson 2004).

However, the results of our studies so far are that the native writers and second language have many error types in common. They may not make the same instances of errors, but the error types are in many cases the same. Both groups make for instance agreement errors, split compounds and violate verb chain patterns. Based on these observations, our starting point have been to not treat second language writers as a special group of writers according to error types. Competence errors made by a second language writer could identical to performance errors of another writer, and vice versa. To judge if an error is made because of lack of grammatical knowledge, or if it was made because of typing failures, is extremely hard to model for a computer program.

If we develop better methods for grammar checking in general, we will develop better methods for grammar checking for second language writers and vice versa. Therefore we started with the method for grammar checking that we already had, namely the grammar checker Granska (Domeij et al. 1999), a state of the art grammar checker based on manually constructed error detection rules.

Our first question was therefore, how is Granska working on texts written by second language writers? Pilot studies were made (Öhrman 2000; Knutsson et al. 2002) and the major problem seemed to be limited coverage of the errors. Granska detected only about 30 % of the errors. Would it then be possible to increase the coverage of Granska without changing the method and technology used? The answer is no. The rule database of Granska was fine tuned with several hundreds of hours of work. If the goal is to increase coverage, the precision of Granska must also be altered. This means lower precision (more false alarms), and we did not believe that it

should be suitable for a user group containing language learners.

Since the error types are too many and too unpredictable we looked for new methods for grammar checking. We developed two statistical methods, ProbGranska and SnålGranska.

ProbGranska (Bigert and Knutsson 2002) detects errors by looking for grammatical constructions that are "different" from known correct text. It detects improbable language constructs using part-of-speech tag trigram frequencies.

SnålGranska (Sjöbergh and Knutsson 2004) requires no manual work, only unannotated text and a few basic NLP tools. The method used is to annotate a lot of errors in written text and train an off-the-shelf machine learning implementation to recognize such errors. To avoid manual annotation artificially created errors are used for training.

In the following sections we will describe these three approaches to grammar checking and show how they perform individually and together on second language learner Swedish texts from the CrossCheck corpus, comparing the results to a commercial Swedish grammar checker.

## 2 The CrossCheck Learner Corpus

The CrossCheck Learner Corpus (or the SVANTE – SVenska ANdraspråks-TExter – Corpus) is a corpus of written second language Swedish. This is a kind of material that has been lacking for Swedish, where the emphasis long has been on the collection and transcription of spoken learner material, such as the EALA/ESFSLD Swedish component[1], a corpus containing spoken conversations and monologues of bilingual school children (Viberg 2001), the ASU (Andraspråkets StrukturUtveckling) Corpus (Hammarberg 1997), and possibly some others. Among these corpora, only ASU has a written component (about 1/3 of the total). Swedish is somewhat unique in this respect, since it is often remarked in the literature on English learner corpora that spoken learner materials are so scarce in comparison to written learner language corpora (e.g. Granger 1998a).

Like ICLE (the International Corpus of Learner English; Granger 1998b; Granger, Hung, and Petch-Tyson 2002) and ASU, the SVANTE Corpus also includes a native speaker part, argumentative essays written as part of Swedish high school national examinations.

A deliberate design feature of the corpus is that it is not intended to be "balanced", at least not by the way we compile it. Rather, we include as much material as we can lay our hands on, including as much relevant

[1] See `http://www.mpi.nl/ISLE/overview/Overview_ESFSLD.html`

metadata as possible, so that users will be able at anytime to extract "virtual corpora" out of the material on the basis of the metadata.

## 3 Granska – A grammar checker using manually constructed rules

For several years we have developed Granska, a spelling and grammar checker for Swedish (Domeij et al. 1999). Granska consists of a spelling checker (Domeij et al. 1994), a part-of-speech tagger (Carlberger and Kann 1999), and about 350 manually constructed rules written in an object-oriented rule language constructed especially for Granska. About half of the rules are error detection and correction rules. An example of a rule detecting agreement errors in a noun phrase is the following.

```
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)
}
```

The first part of the rule detects the agreement error. The second part tells what should happen after a matching. The `mark` statement specifies that the erroneous phrase should be marked in the text, the `corr` statement that a function is used to generate a new inflection of the article from the lexicon, one that agrees with the noun. This correction suggestion is presented to the user together with a diagnostic comment (in the `info` statement) describing the error.

The rules are compiled and optimized using statistics of words and tag bigrams in Swedish. This means that each rule is checked by the matcher *only* at the positions in the text where the words or tag bigrams of the least probable position in the rule occur.

A subset of the rules of Granska constitutes a shallow parser for Swedish, called GTA (Knutsson et al. 2003).

## 4 ProbGranska – A statistically based grammar checker

ProbGranska is a probabilistic algorithm for detection of context-sensitive spelling errors (Bigert and Knutsson 2002). The algorithm is divided into two parts: a statistical part and a transformation part.

### 4.1 Statistical information

The first, statistical part of the algorithm uses PoS tag trigram frequency information, gathered from a corpus of the target language. The general observation is that a grammatical construction is probably malformed if it contains previously unseen trigrams. Unfortunately, human language is very productive, and new, unseen grammatical constructs will arise. To address this problem, we broaden the concept of a PoS tag trigram.

Two PoS tags that are used in similar syntactic contexts are said to be close. We want to use this closeness, or *distance*, between tags to mitigate the effect of rare trigrams due to the productivity of the language.

We calculate the distance between two tags by using the frequencies of PoS tag trigrams obtained from the corpus. Given two tags $t$ and $r$, we look up the frequencies for the trigrams $(t_1, t, t_2)$ and $(t_1, r, t_2)$. Naturally, if either $t$ or $r$ is more frequent than the other, the trigram frequencies will be higher and thus, we have to compensate for the tag frequencies. We obtain $P_t(t_1, t_2) = \mathit{freq}(t_1, t, t_2)/\mathit{freq}(t)$.

From this, we can apply a number of similarity measures from the work of Lee (1999), e.g. the L1 norm: $L1(P_t, P_r) = \sum_{t_1,t_2} |P_t(t_1, t_2) - P_r(t_1, t_2)|$, where the sum is over all tag pairs $t_1, t_2$ in the tag set. We see that the distance increases when the trigrams differ in frequencies. The measure will give us a list of similarities between every pair of tags $t$ and $r$. Suitably normalized the values can be used as probabilities. Thus, if $p$ is $L1(P_t, P_r)$ normalized (i.e. the distance between $t$ and $r$), $p$ can be seen as the probability of retaining grammaticality when replacing a word having PoS tag $t$ with a word having PoS tag $r$.

Now, given a rare trigram $(t_1, t, t_2)$, we attempt to replace one (or more) of the tags with another tag close in distance. For example, if replacing $t$ with $r$, we obtain $(t_1, r, t_2)$. To penalize the tag change, we multiply the frequency of the new trigram with the probability $p$ of the tag change. Now, if the penalized frequency is high (as defined by an arbitrary threshold $e$), the grammatical construction is most probably correct and the low frequency was originally due to a rare tag. If all attempted tag replacements result in low frequencies, the trigram is probably not grammatical and is marked as an error.

### 4.2 Phrase transformations

Most false alarms occur near the beginning or end of a phrase constituent. There, a trigram covers two phrases and the productivity of the language gives rise to almost any combination of PoS tags. Furthermore, rare phrase constructions often produce rare PoS tag trigrams. Normally, the simplest (or shortest) form of a phrase is the most common, e.g. *the men* is a more common type of NP than *the little green men*. Thus, when faced with a potential error as described in the previous subsection, we will identify all adjacent phrases and try to simplify. Hopefully, the rare trigram is due to a rare combination of phrases.

We attempt to transform a phrase to a simpler form by replacing it with a more common phrase of the same type. To this end, we use GTA, the shallow parser of Granska. Since we try to retain the inflectional information of the phrase, the new sentence will most probably be grammatical. For example: an error is detected near the words *are old are* in *All paintings that are old are for sale*. The NP *all paintings that are old* is reduced to *the paintings* and the sentence becomes *The paintings are for sale*, avoiding the rare construction.

The sentence resulting from the phrase transformation is fed to the algorithm in the previous section. If the new sentence is not erroneous, it is probably grammatically correct. If all phrase transformations fail (are reported as errors), there is probably a grammatical error and this is reported to the user. Rare PoS tag trigrams also occur frequently near a clause beginning or end. We decided not to look for errors in trigrams that cross a clause boundary. Hence, the largest unit is not a sentence but a clause.

### 4.3 Evaluation

The procedure used to evaluate the error detection algorithm is fully automated and requires no resources annotated with errors (Bigert 2004). Furthermore, it is portable to any language and tag set (given a dictionary in that language) and produces reproducible evaluations. The idea behind the procedure is to introduce artificial spelling errors into error-free text. A graph of precision versus recall is shown in Figure 1 where 2% errors were introduced into the text. As seen from the figure, the proposed method increases the precision significantly while sacrificing recall.

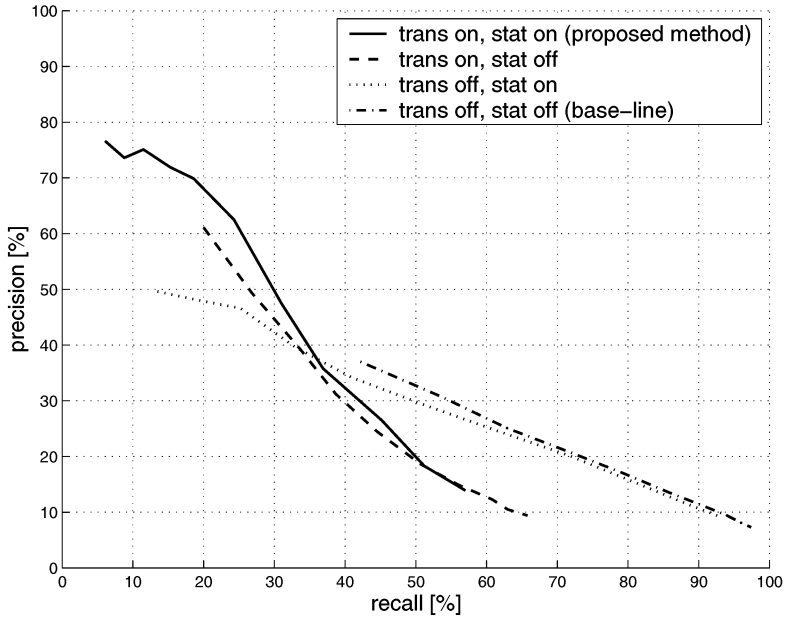


Figure 1: *Precision and recall of ProbGranska at the 2% error level.*

## 5 SnålGranska – A grammar checker requiring no manual work

The third method, SnålGranska (Stingy Checker), is based on machine learning of automatically constructed errors (Sjöbergh and Knutsson 2004). The main strength of SnålGranska is that almost no manual work is required to create it. It also has modest requirements on the NLP tools it uses. Another advantage is that the method is based on how correct language use looks, though not as much as ProbGranska. This means that it is able to detect errors that are hard to describe by manually written rules.

Currently, statistical methods are used for a wide range of tasks in the NLP area. One way to use this approach for grammar checking would be to treat it as a tagging task. First, collect a lot of text. Annotate all errors with ”ERROR” and the correct text with ”OK”. When this is done, train an off-the-shelf machine learning implementation on this annotated data. It will now be trained to detect errors in text.

This approach has two drawbacks. These methods generally require quite a lot of data, and finding enough text with unintentional errors in could be a problem. The largest drawback though is that a lot of manual work is required to find all errors and annotate them.

SnålGranska avoids this problem by using artificial errors. First, a lot of unannotated and (mostly) correct text is collected. Then the text is cor-

rupted by inserting simple errors automatically. Since these errors are inserted automatically, they can be annotated automatically at the same time. Then a machine learning algorithm is trained on this data and applied to new texts as a grammar checker.

The artificial errors that are generated can be of any type. As the strength of SnålGranska is the minimal amount of manual work required, only very simple errors have been used. Two errors types have been tested, split compound errors and agreement errors. Split compounds is an example of an error type that SnålGranska is well suited for. It is easy to split compounds, but the resulting sentence structure from split compounds is in general hard to predict, so it is hard to write manual rules for these errors. We use a modified spelling checker (Domeij et al. 1994; Kann et al. 2001) and split all compounds recognized by the spelling checker.

Agreement errors is an error type which SnålGranska is less suited for. It is the most thoroughly covered error type in current grammar checkers for Swedish, and it is relatively straightforward to write good rules for these manually. To see how well the SnålGranska method works compared to state of the art grammar checking, this was also tested. Errors were generated by randomly choosing words with gender, number or definiteness features, and replacing them with another word with the same lemma but another surface form, using a simple dictionary lookup.

Using more ”human like” artificial errors should be expected to produce a better grammar checker, but requires more work. If a lot of time is spent on producing a sophisticated error generation program, this time could perhaps have been spent better by writing rules for a traditional grammar checker. As long as the resulting grammar checker is useful, the simpler the error generation the better. For the examples above, about 15 minutes have been spent on error generation. This is the only manual work required, everything else is done automatically.

Almost any machine learning algorithm could be used for SnålGranska. We use fnTBL (Ngai and Florian 2001), which produces rules that are easily understood by humans. As features for the machine learner we use words and their part-of-speech, which is automatically assigned by a tagger. For split compounds we also use the spelling checker to filter out false alarms, by checking if the (suspected) errors combine into acceptable words. No other resources are needed.

When an artificial error results in a sentence that is also correct, which is quite possible, it will still be an error to the machine learning algorithm. This is not a problem, since there is also a lot of examples of correct language use in the training data. This means that (generally) only the properties of those artificially inserted errors that result in sentences that are not

| | MS Word | Granska | Prob-Granska | Snål-Granska | Any Granska | Total |
|---|---|---|---|---|---|---|
| All detected errors | 392 | 411 | 102 | 121 | 528 | 592 |
| All false positives | 21 | 13 | 19 | 19 | 48 | – |
| Detected spelling errors | 334 | 293 | 35 | 26 | 314 | 363 |
| False positives | 18 | 5 | – | – | 5 | – |
| Detected gram. errors | 58 | 118 | 67 | 95 | 214 | 229 |
| False positives | 3 | 8 | 19 | 19 | 43 | – |

Table 1: Evaluation on second language learner essays, 10 000 words. Any Granska means all errors detected by any of the Granska methods.

correct will be learned.

SnålGranska could be used on many error types, but so far only these two have been tested. This means that many easily detectable error types, such as repeated words or wrong verb tense after the infinitive marker *att*, are ignored by SnålGranska, which leads to low overall recall.

## 6 Evaluation

To evaluate the different grammar checking methods we used essays written by people learning Swedish as a second language. These were taken from the SSM-corpus part of the CrossCheck corpus. These texts contain a lot of errors, which is generally good for the grammar checkers (easier to get high precision), but it also leads to problems for the grammar checkers. Many errors overlap, which can give unexpected results. There is also often very little correct text to base any analysis on.

All methods were run on about 10 000 words of text from the essays. The grammar checker for Swedish in Microsoft Word 2000 was also run on the same text, as a comparison to other available methods. The grammar checker in MS Word has been developed for high precision, while for instance SnålGranska was developed for high recall (on the two error types it detects). All alarms from the grammar checkers were manually checked to see if there was a true error or a false alarm. Results are shown in Table 1. The texts were not manually checked to find all errors, but a manually checked sample shows that many errors go undetected. Less than half of the errors in the sample were detected.

The grammar checkers using manually constructed rules, Granska and MS Word, show higher precision (about 95%) than the other methods (about 85%). They also detect many more errors, mainly because they also look for spelling errors, which are common and much easier to detect. When it

| Pair | | Both | Only Granska | Only Prob-Granska | Only Snål-Granska | Any |
|---|---|---|---|---|---|---|
| Granska+ProbGr. | Correct | 17 | 101 | 50 | | 168 |
| | False alarms | 0 | 8 | 19 | | 27 |
| Granska+SnålGr. | Correct | 44 | 74 | | 51 | 169 |
| | False alarms | 3 | 5 | | 16 | 24 |
| ProbGr.+SnålGr. | Correct | 11 | | 56 | 84 | 151 |
| | False alarms | 0 | | 19 | 19 | 38 |

Table 2: Pairwise overlap in detection of grammatical errors between Granska, ProbGranska and SnålGranska.

comes to grammatical errors the recall is similar for all methods.

While the manual rules of Granska detect more errors, and with higher precision, than the other methods, it still misses many errors detected by other methods. ProbGranska in particular was developed explicitly to find errors which are hard to detect using manual rules. In Table 2 the pairwise overlap in detections of grammatical errors for the different methods developed in the CrossCheck project is shown. The three methods complement each other and by combining them much better coverage can be achieved.

Even SnålGranska, which currently is only trained on split compound errors and agreement errors, two error types already covered by Granska, finds many errors that the other methods do not detect. It could also be extended with more error types, for improved recall.

Combining the different methods could be done in many ways. One simple method is to treat any detection from any method as an error. In Table 1 the results using this method are shown.

## 7 Discussion

The evaluation of our three approaches to grammar checking in Section 6 showed that the three methods to a large extent detect different errors and therefore complement each other well. We therefore propose that a grammar checker should combine different approaches to grammar checking.

How can a grammar checker be further improved to detect even more of the errors? All three methods described in this chapter rely on the same type of part-of-speech disambiguation. The main problem is that grammatical errors sometimes are misinterpreted as correct grammatical constructions. Independent of which grammar checking method that is used after word class disambiguation, many errors cannot be detected. Our initial studies

showed that word class disambiguation is necessary to limit the amount of false alarms. What we need is a language model that is much more rigid than the current model. This is a veritable case of Heller's Catch 22, a rigid language model would not analyse ill-formed constructions at all, and we are thereby back into the deep parsing dilemma – where many sentences are not parsed either because they are ungrammatical or because of limitations of the current grammar. The problem of the general analysis of ungrammatical constructions is one of the main bottlenecks for further improvements of current methods for grammar checking.

The methods for grammar checking described in this chapter are already integrated (Granska and ProbGranska) or close to be integrated (SnålGranska) into a language-learning environment called Grim[2]. Grim is a web client with basic word processing facilities, which is connected to several network based language tools (bilingual lexicons, a grammatical analyzer, a word inflector, a interface to a concordancer).

In the design of Grim it has been important to provide the user with several different views of language. For the case of grammar checking, Granska represents a rule-based view of language, ProbGranska a more statistical view, and SnålGranska is something in between. The idea and the contribution of using three methods for grammar checking, beside increased coverage and accuracy, is to make the user aware of how different tools can give different feedback on the user's writing, and that different linguistic resources will treat language in different ways. One pedagogical problem is how to explain for the user that three methods are better than one. A second pedagogical problem is how to show that the three methods co-exist in an environment with several language tools seamlessly integrated. One important kind of feedback that we have got from the users of Grim so far, is that they view Grim as one program, not as an interface to several different language tools and programs.

**Acknowledgements**

We are grateful to Lars Borin, Janne Lindberg and Gunnar Erikssson for their work on the CrossCheck corpus, and to Stefan Westlund, who developed the Grim interface to Granska and ProbGranska.

[2]See `http://skrutten.nada.kth.se`

## References


Bigert, J. 2004. Probabilistic detection of context-sensitive spelling errors. In *Proc. 4th Int. Conf. Language Resources and Evaluation (LREC 2004)*.

Bigert, J. and Knutsson, O. 2002. Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proc. 2nd Workshop of Robust Methods in Analysis of Natural Language Data*, pp. 10–19.

Bolt, P. 1992. An evaluation of grammar-checking programs as self-help learning aids for learners of English as a foreign language. *Computer Assisted Learning 5*(1–2), 49–91.

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

Chen, J. F. 1997. Computer generated error feedback and writing process: A link. *TESL-EJ Teaching English as a second Foreign Language 2*(3).

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

Domeij, R., Knutsson, O., Carlberger, J., and Kann, V. 1999. Granska – an efficient hybrid system for Swedish grammar checking. In *Proc. 12th Nordic Conf. on Computational Linguistics*.

Granger, S. 1998a. The computer learner corpus: A versatile new source of data for SLA research. In S. Granger (Ed.), *Learner English on computer*, pp. 3–18. London: Longman.

Granger, S. (Ed.) 1998b. *Learner English on Computer*. London: Longman.

Granger, S., Hung, J., and Petch-Tyson, S. (Eds.) 2002. *Computer learner corpora, second language acquisition and foreign language teaching*. Number 6 in Language Learning and Language Teaching. Amsterdam: John Benjamins.

Hammarberg, B. 1997. *Manual of the ASU corpus, a longitudinal text corpus of adult learner Swedish. Version 1997–04–10*. Stockholm University, Department of Linguistics.

Kann, V., Domeij, R., Hollman, J., and Tillenius, M. 2001. Implementation aspects and applications of a spelling correction algorithm. In

L. Uhlirova, G. Wimmer, G. Altmann, and R. Koehler (Eds.), *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, Volume 60 of *Quantitative Linguistics*, pp. 108–123. Trier, Germany: WVT. Available on the web from `http://www.nada.kth.se/theory/projects/swedish.html`.

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

Knutsson, O., Pargman, T. C., and Eklundh, K. S. 2002. Computer support for second language learners' free text production – Initial studies. In *Proc. 5th Int. Workshop on Interactive Computer Aided Learning*.

Lee, L. 1999. Measures of distributional similarity. In *Proc. 37th Annual Meeting of the ACL*, pp. 25–32.

Lindberg, J. and Eriksson, G. 2004. CrossCheck-korpusen – en elektronisk svensk inlärarkorpus. In *Proc. ASLA 2004 Conference*.

Ngai, G. and Florian, R. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, Carnegie Mellon University, Pittsburgh, USA, pp. 40–47.

Park, J. C., Palmer, M., and Washburn, G. 1997. An English grammar checker as a writing aid for students of English as a second language. In *Proc Conf. on Applied Natural Language Process.*

Sjöbergh, J. and Knutsson, O. 2004. Faking errors to avoid making errors: Very weakly supervised learning for error detection in writing. In preparation.

Vernon, A. 2000. Computerized grammar checkers 2000: capabilities, limitations, and pedagogical possibilities. *Computers and Composition 17*, 329–349.

Viberg, Å. 2001. Age-related and L2-related features in bilingual narrative development in Sweden. In L. Verhoeven and S. Strömqvist (Eds.), *Narrative development in a multilingual context*, pp. 87–128. Amsterdam: John Benjamins.

Yazdani, M. 1990. An artificial intelligence approach to second language learning. *J. Artificial Intelligence in Education 1*, 85–90.

Öhrman, L. 2000. Datorstödd språkgranskning och andraspråksinlärare. Technical report, Institutionen för lingvistik, Stockholms Universitet. D-uppsats i datorlingvistik.

# Paper 5

# Designing and Developing a Language Environment for Second Language Writers

Ola Knutsson[*1], Teresa Cerratto Pargman[2], Kerstin Severinson Eklundh[1] and Stefan Westlund[1]

## Abstract

This paper presents a field study carried out with learners who used a grammar checker in real writing tasks in an advanced course at a Swedish university. The objective of the study was to investigate how students made use of the grammar checker in their writing while learning Swedish as a second language. Sixteen students with different linguistic and cultural backgrounds participated in the study. A judgment procedure was conducted by the learners on the alarms from the grammar checker. The students' texts were also collected in two versions; a version written before the session with the grammar checker, and a version after the session. This procedure made it possible to study to what extent the students followed the advice from the grammar checker, and how this was related to their judgments of its behavior.

The results obtained demonstrated that although most of the alarms from the grammar checker were accurate, some alarms were very hard for the students to judge correctly. The results also showed that providing the student with feedback on different aspects of their interlanguage, not only errors, and facilitating the processes of language exploration and reflection are important processes to be supported in second-language learning environments.

Based on these results, design principles were identified and integrated in the development of Grim, an interactive language–learning program for Swedish. We present the design of Grim, which is grounded in visualization of grammatical categories and examples of language use, providing tools for both focus on linguistic code features and language comprehension.

Keywords: adult learning; evaluation of CAL systems; human-computer interface; interactive learning environments.

## 1 Introduction

Technology has the potential to play a large role in the process of learning a second language (Warschauer, 1996; Warschauer & Meskill, 2000). However, the

[*] Corresponding author, knutsson@nada.kth.se, fax: +46-(0)8-10 24 77.
[1] School of Computer Science and Communication, Royal Institute of Technology, SE-100 44 Stockholm, Sweden. {knutsson, kse, d98-swe}@nada.kth.se.
[2] Computer and Systems Science, University of Stockholm, Forum 100, SE-164 40 Kista, Sweden. tessy@dsv.su.se.

development of this potential is still in the early stages. Issues on which the realization of this potential depends include "the shift from thinking of technology as assisting instruction to thinking of it as supporting and facilitating learning" (Garrett, 1991, p.95). On the one hand, computer programs for learning language should be able to understand a user's input and evaluate it not just for correctness but also for appropriateness. On the other hand, "the use of the computer does not constitute a method" (Garrett, 1991. p. 75). Rather, the computer is a tool, an instrument, in which a variety of methods, frameworks, and pedagogical philosophies may be integrated and implemented. The usefulness of computer assisted language learning cannot reside in the medium itself but in how it is put to use.

The work presented in this paper is part of a project focusing on the use of computer-based language tools for language learning. In particular, we are interested in:

- studying how learners develop their writing practices in the context of learning Swedish as a second language. How do they use available writing tools and how do these tools shape learners' understanding of the new language?
- contributing to the improvement of the design and development of existing language tools for writing in learning contexts.

We believe that the study of these questions will contribute to a better understanding of the role of computer-based language tools in complex processes such as second language learning and that this will help developers to better understand what is at stake when designing for learning purposes.

## 1.1 Computers in second language learning

Computers have been used in second language teaching for about twenty-five years (cf. Levy 1997). However, they have never been recognized by the majority of language teachers as exemplifying good teaching, and still remain peripheral to the core of classroom teaching (Laurillard and Marullo, 1993). These programs have often been either based on behaviorist or cognitive models of learning (see Warschauer and Meskill, 2000).

In contrast to these approaches, communicational approaches emphasize the social aspect of language acquisition and view learning language as a process of apprenticeship or socialization into particular discourse communities (cf. Warschauer and Meskill, 2000). Furthermore, with the arrival of the Internet, communicational approaches have a powerful medium for assisting language learning. According to Warschauer and Meskill (2000), there are today many ways in which students and teachers can use the Internet to facilitate target language interaction (cf. computer-mediated communication in a classroom; computer-mediated interaction for long distance exchange; accessing resources and publishing on the World Wide Web).

However, as Laurillard and Marullo (1993) argued, communicational approaches are not a panacea. When communication breaks down it is often caused by grammatical problems, which introduce unclear language.

Without turning back to traditional learning approaches with for instance isolated exercises on grammatical forms to improve the learner's grammatical knowledge a

new approach called focus on form was proposed by Long (cf. Long & Robinson, 1998). Focus on form means to draw the learner's attention to linguistic code features while conducting meaningful tasks (i.e. achieving communicational goals). What kind of form to focus on and how this should be done are questions that have to be carefully treated by developers of computer programs as well as teachers using them in education.

We suggest that providing tools to support analysis, reflection and exploration of learners' writings as well as text material available in the target language, will provide the learner with explicit[3] and implicit feedback. In that respect, we agree with Warschauer and Healey (1998) that we should not hope for human-like intelligence in the feedback that a computer assisted language-learning program can provide. Instead we should use the power of the computer for easy and meaningful interaction with the learning material including feedback and guidance, supporting different learning styles as well as enabling communication between language learners and users in multimodal environments.

In this sense, it is essential to provide adult learners with tools for helping them to develop understanding, progress and enjoyment when writing in the target language. In the work presented in this paper, user studies were conducted with the purpose to investigate the role of language technology in the development of second-language learning and the specific activities that should be supported.

## 1.2 A developmental perspective on the use of language tools

Our interest in supporting writing relies on the central place that writing occupies in the development of language and thinking processes (Vygotsky, 1962; 1978). Writing does not only allow one to do new things but more importantly, it brings linguistic categories into awareness (cf. Olson, 1995; Lantolf 2000). From this perspective, writing mediated by the writing system is of utmost importance as it affects consciousness and cognition through providing a model for language.

We regard writing as a base for hypothesis testing in the light of Swain's output hypothesis (Swain, 2000), that speaking and writing in the second-language push the learner to use language more deeply and thereby become more aware of grammatical forms and their meaning when achieving communication goals.

An important question regards the role of language tools in supporting learning, and more in particular, in helping learners to reflect on and develop awareness of the language they produce. According to Säljö (1996), the role of tools – psychological as well as technical – and the concept of mediation play a fundamental role in the understanding of human thinking and learning. From this perspective on language and tools, the use of language tools may alter writing processes. Our inquiry entails examining not only the transformative power of tools on developmental socio-cognitive processes, but also how the computer-based language tools are developed, and how they get transformed by the users (cf. Verillon & Rabardel, 1995; Cerratto, 1999).

---

[3] By explicit feedback we mean the feedback that is given in the form of lexical and grammatical knowledge. By implicit feedback we mean the feedback that represents the form of evidence of language use in the target language.

## 1.3 Computer language tools in language learning

Most computer-assisted language learning (CALL) systems that are used today cannot process unconstrained language use. Computer programs being able to make some more advanced processing in limited domains like those called intelligent language tutors (cf. Levy, 1997) are left out in our work. We are mostly interested in computer tools capable to some extent, to analyze learners' written language and whose focus is mostly on the feedback provided on unconstrained language use. In other words, our focus is on these kinds of computer programs that can give feedback on the learner's free language production.

A technology that has the potential for a task of this kind is language technology. Language technology (LT) or computational linguistics has so far not been part of mainstream CALL. According to Levy (1997), and Chapelle (2001) computational linguistics seems to concern mostly syntactic parsing; however we think that this definition is very limited. We regard LT as a broad field, with the potential to support different levels of language use and different processes in language learning. Nerbonne (2002) presents a good overview on how LT is used to today in CALL and how it can be used in the future.

Grammar checkers and other language tools have been designed for second language learners (mostly for English as a second language), for some time now. There exist commercial grammar checkers for second language writers (e.g. NativeEnglish) as well as several research prototypes (Park et al, 1997; Bolt and Yazdani 1998; Schneider & McCoy, 1998; Menzel & Scroeder, 1999; Izumi et al, 2003; Vandeventer Faltin, 2003; Bender et al 2004). A language tool that includes more functionality than just a grammar checker is presented by Cornu et al (1996). This tool also includes a verb conjugator, a set expression translator, a French-English bilingual dictionary, an on-line grammar and a list of difficult words for French speakers writing in English.

## 1.4 Errors and grammar checking programs

At first view, our interest in learner's language errors may seem strange: why should we focus on what learners get wrong rather than on what they get right? We have good reasons for focusing on errors. From a developmental perspective, learners' errors are a rich source for understanding how they make sense of and construct a new symbolic system. In the words of James (1998) "The learners' errors are a register of their current perspective of the target language" (p. 7). Errors can also be viewed as the expression of a conflict between the learner's conceptions of what is the correct use of the target language and what is really correct use in the target language. Reflecting on learners' misconceptions is a way to understand their errors as steps in a developmental process. From this perspective, "the pedagogic emphasis is placed instead on enlarging the individual learner's awareness of written English as a resource for making meaning in the student's particular field of study" (Scott 2001, p.164). A developmental perspective on learners' grammatical errors has important implications on the way adults develop their interlanguage – the new symbolic system – reconstructing it from their understanding, needs and already acquired language resources. Helping the learners to notice grammatical forms (in our case errors) might work as a "notice the gap principle", which means that learners become aware of the gap between their interlanguage and the target language (Swain, 2000).

Grammar checking programs are designed to detect, diagnose, and correct grammatical errors in written texts. But the way they do it seems to be limited in the light of James's error taxonomy. According to James (1998), there exist different classes of deviance:

- *Grammaticality*: an objective grammar of the target language "decides" whether some bit of language is grammatical or not.
- *Acceptability*: native speakers (individually) "decide" whether some bit of language is acceptable or not.
- *Correctness*: speakers influenced by prescriptive standards "decide" whether some bit of grammatical and/or acceptable language is correct or not.
- *Strangeness*: some bit of language that is ungrammatical from the learner's point of view, but used and acceptable when used by for instance poets or other language acrobats.
- *Infelicity*: errors or gaps of learner's repertoire with socio-linguistic consequences when performing different speech acts.
  (James, 1998, p. 64)

This view of deviance concerns the relation between the learner's target language output and the grammar of the target language, its users, its prescriptive normative standards, co-occurrence restrictions and the performance of language-specific speech acts. The deviance can also be viewed in the opposite direction, from the learner's errors to the learner's own interlanguage grammar. James (1998) makes the following classification of what kind of errors learners may produce:

- *Slips*, are lapses of tongue, pen or fingers on the keyboard, these can quickly be detected and self-corrected.
- *Mistakes*, can only be corrected by the user if the deviance is pointed out to him or her. If a simple indication of the deviance in the text is sufficient for self-correction, it is a first order mistake. If further information is needed, in the form of an exact location or diagnosis, it is a second order mistake.
- *Errors*, cannot be self-corrected, and involve language constructions that are either allowed in the learner's "grammar" or not covered by it. Errors require further relevant learning before they can be self-corrected.
- *Solecisms*, break the rules of correctness as laid down by purists. An example from Swedish is the ambiguous interpretation of prepositions and conjunctions as in *större än mig/jag* (eng. *bigger than me/I*). Solecisms are easy to teach and learn, but they are not important for the learners' interlanguage development or conversion to the target language norm.

This classification is important for the understanding of the functionality of a grammar checker, and its consequences and needs to give adequate feedback. It is also useful when deciding the content of language learning systems. Slips and mistakes might only need implicit feedback, but errors need explicit feedback, for instance using specially designed learning material of the current language construction. An interesting question is: is there any point in the learner detecting and correcting slips? Or is the program a good help to get rid of the low-level errors that learners may produce when revising a text?

## 1.5 Role of written feedback

Several researchers have studied the impact of teachers' written feedback. For example, Laurillard and Marullo (1993) argue that explicit feedback is essential for the learner's language understanding. Feedback in their terms "must deal explicitly with any diagnosed misconceptions or mal-rules, must provide help with analyzing the gap between learner performance and target form in terms of the information supplied, or in terms of previously mastered items" (p. 157). However, there is debate about the role of feedback on errors. Some researchers argue that it is harmful (Truscott, 1999) while most others suggest that feedback on errors is helpful despite the fact that it seems to be hard to get strong scientific evidence for the relevance of feedback (Ferris, 2004).

James (1998), presenting a six-level model of feedback on errors suggests that the first step in error feedback is *error detection*. Error detection only means to signal that the sentence or segment is erroneous. This feedback makes the learner aware of the presence of an error. This might be enough information for the learner if the error is a slip or a first order mistake. If it is a second order mistake, the teacher or the program must locate the error in the paragraph. This is called *error location*. Error location can be straightforward or more problematic. Some errors are easy to locate, e.g. agreement errors in Swedish, in the sentence *Jag såg en hundar* (eng. *I saw a dogs*). But others are more difficult, like a missing finite verb or a missing subject, where it is more problematic to locate where the missing word should have been placed in the sentence.

When the error is located, it can be described in linguistic terms and the teacher or the program makes an *error description*. A very important question is raised when this kind of feedback is to be given; in which language should the error description be written? Is it most fruitful to make the descriptions in the mother language or the target language?

The description should be used both by the learners and the teachers. In this context, the language has to be self-explanatory, simple and easily learned. To describe the errors in detail we also need a system that is well elaborated and powerful in its description, because all language users make advanced errors (James, 1998, p. 97). Therefore, a framework for error classification and categorization is needed. The next step is *error diagnosis*, to explain why the learner has produced the error. Closely related to error diagnosis is *error explanation* – how the construction in the interlanguage differs from the correct construction in the target language.

The last part of the feedback on the learner's errors is *error correction*, proposing specific corrections. Is error correction relevant feedback? James' conclusion is that error correction works (James, 1998). Correcting grammar errors both improves the grammar and the content in the rewritten text (Fathman and Whalley, 1990). Moreover, several studies show that students want to be corrected, and as James points out: why neglect what the learners want?

# 2 Empirical studies on the use of a Swedish grammar checker by second-language learners

We initiated our studies with a pilot study, the aim of which was to develop methods for the study of the use of a Swedish grammar checker, Granska, in second language learners' free text production (Knutsson et al, 2002). After this, a field study at a Swedish university was carried out with the same tool and methodology. Its first part was reported in (Knutsson et al, 2003), and its entirety is reported in the following text.

The aim of the studies was:

- to understand the impact of the spell- and grammar checker Granska on the users' texts, as well their way of adapting the grammar checker to their writing purposes.
- to guide the redesign of the grammar checker Granska and the design of a new language-learning environment.

## 2.1 Granska – a grammar checker for Swedish

Granska is a grammar checker for Swedish developed at the Royal Institute of Technology in Sweden (Domeij et al, 2000). Granska combines probabilistic and rule-based methods to achieve high efficiency and robustness. Using special error rules, the system can detect a number of Swedish grammar problems and suggest corrections for them that are presented to the user together with instructional information. The core of the Granska system is a statistical word class analyzer, a collection of phenomena-based grammar checking rules, and a robust shallow parser. Granska was designed for writers with Swedish as a first language. However, although first and second language writers might make very different kinds of errors, they also have many error types in common. Therefore we decided to study the use of Granska in second language writing.

In the studies presented in this paper, Granska was used with a simple web interface running in a batch mode. The user interface allowed the students to use the word processor they preferred on any platform for writing, but on the other hand, they had to switch between two screen windows in order to use the feedback from Granska when revising their text in the word processor.

Granska has been textually evaluated on five different text genres including mostly texts from native speakers of Swedish (Domeij et al, 2002). The accuracy of Granska depended much on the text genre. A recent comparison with the Swedish grammar checker in Microsoft Word 2000 (Birn, 2000) showed that Word's grammar checker caused fewer false alarms, but also missed more errors than Granska (Bigert et al, 2004). A false alarm is when the program incorrectly points out a grammatical piece of language as ungrammatical. The spelling checkers of each program showed the opposite results. The evaluation was made on 10 000 words written by second language learners of Swedish taken from the SSM-corpus part of the CrossCheck learner corpus (Lindberg & Ericsson, 2004). A manually checked sample showed that more than 50% of all errors were undetected by both grammar checkers.

## 2.2 Participants in the study

We observed a group of second-language learners having resided in Sweden from 3 months to 10 years. The group consisted of 16 students who participated voluntarily. They were in their third and final semester of the language-learning program ”Svenska som främmande språk” (Swedish as a foreign language). This program prepares learners to pass the Tisus test, which they have to pass in order to be qualified for entering Swedish university courses.

The participants wrote several texts as part of the writing course in Swedish as a second language. The texts represented different genres: argumentative texts, letters, descriptions, and essays, and they covered different topics. Participants composed all texts at home and discussed them at the university. The teacher reviewed their texts and graded them.

Participants came from different parts of the world, with mixed backgrounds, and had different degrees of familiarity with the Swedish language. Eleven mother languages were spoken in the group: Belrussian, Dutch, Farsi, Finnish, Hungarian, Japanese, Polish, Quechua, Romanian, Russian and Spanish. Two participants were bilingual. Only one participant was male. The participants were between 20 and 40 years old.

## 2.3 Methodology

To study the use of a language technology tool in a real setting entails, first of all to introduce the tool into the new context, establishing and developing a relationship with the teacher and the students, and gaining the confidence of all participants. We started our study by presenting the study to the students together with the teacher.

In a first meeting at the university we carefully explained the tool Granska and emphasized that Granska is a computer tool with limited knowledge of language. We gave them both oral and written instructions describing their role and tasks in the study. The instruction mostly concerned how they should work with Granska, how they should deliver and save their texts and in what form.

Participants were encouraged to use Granska outside the class. The instruction for the participants was the following: “use the Granska whenever you want and when you feel it will help you”. Some of the control of the data collection was thus left to the participants. We arranged several meetings with the participants to secure that they understood their tasks.

## 2.4 Data collected

After the first meeting we introduced Granska in the computer laboratory, with the purpose to make sure that participants had understood our instructions and to make observations of their interaction with the computer, Granska and their teacher in this new setting. These observations were made on group basis, and were documented by note taking.

An important part of our study is the collection of the participants’ original text versions and the final versions after the use of Granska. One purpose of this collection was to make it possible for us to trace the effects of Granska’s alarms. One question was: In what way did the participants change their text according to the feedback

from Granska? We were especially interested in what happened when a false alarm occurred, i.e. did the participants understand the false alarms as a false alarm or not?

We conducted interviews with the participants after they had used Granska on their own. The questions asked during were based on an interview guide. The users were interviewed in small groups in a semi-structured way. The interview guide contained questions about the use of other grammar checkers, and their opinion about them. Other questions focused on the use of Granska and its current interface, its meta-language, and if they thought that Granska is useful for someone learning Swedish as a second language. We also asked questions about new possible functionality (e.g. new language tools, or detection of other error types), and what the students thought is important in a program for someone learning Swedish.

In order to get detailed information about specific alarms from Granska we instructed the users to judge the alarms from Granska. They were instructed to judge the feedback components of every alarm. To make the procedure easier for the users we had used a three-level model for the feedback to be judged, namely the detection, diagnosis and the correction proposal(s) from Granska, see Figure 1. The term *detection* covers both James' (1998) terms *error detection* and *error location*, the term *diagnosis* covers James' terms *error description* and *error diagnosis*. The term *correction* is the same as James' term *error correction.*

This judgment procedure has some similarities with the method "grammaticality judgment" frequently used in SLA research (cf. Gass 1994; Goss et al 1994). One difference is that the users only judge their own language that has been pointed out as ungrammatical by Granska. The procedure involved judgments of Granska's view of grammaticality, acceptability, and correctness. We used a graded scale because we believed that Granska's feedback could be judged useful/not useful on a continuum.

The users were instructed to use the following grading scale for their judgments:

- Grade 5. Excellent – *I understand exactly what Granska suggests.*
- Grade 4. Good – *Granska is quite a good help for me.*
- Grade 3. Acceptable – *It is hard for me to make up my mind on what Granska says, but I take a chance that Granska is right.*
- Grade 2. Bad – *It is hard for me to make up my mind on what Granska says, I have to look in my grammar book. With the help of the book I can decide if I should follow Granska or not.*
- Grade 1. Incomprehensible – *I do not understand what Granska says. I have to ask the teacher or some other competent person for help.*

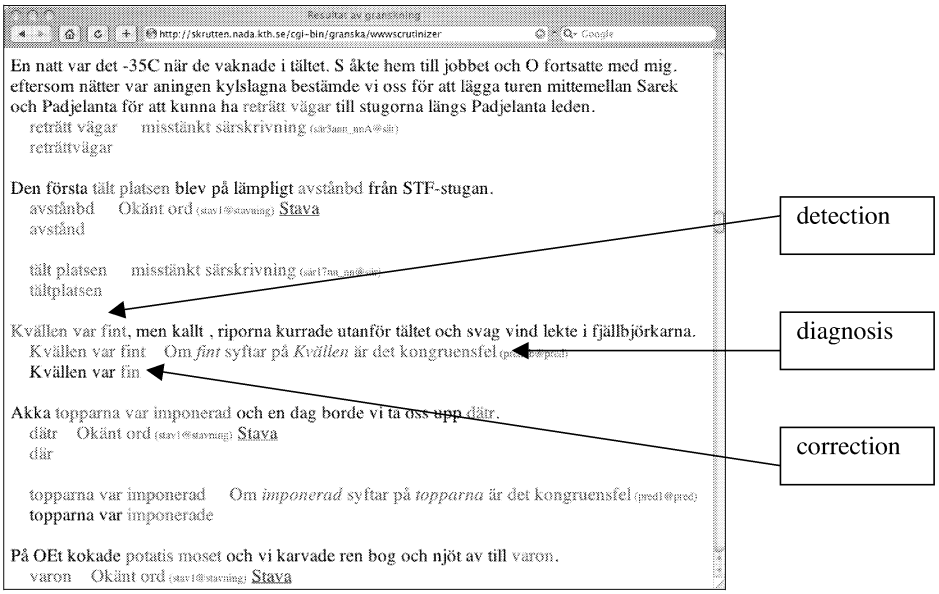


Figure 1. *The output from Granska as an html-page viewed in a web browser. The arrows point out what was meant with the feedback components detection, diagnosis and correction in the instruction to the users. Colors were used to separate the feedback components from each other. The screenshot was used in the instruction to the users.*

The participants could make their judgments electronically by editing an html page generated by Granska, but judgments on printouts was also possible. We also instructed them to make free text comments when they felt that the grades were not enough or if they wanted to make more general comments about Granska's behavior.

# 3 Results

In order to capture the users' free writing and interaction with Granska the data collection was partly unsupervised. The collection of texts, final versions and judgments was based on the users willingness to take part in the study. This resulted in different levels of commitment to the given instructions. Table 1 shows the participants that made judgments when using Granska.

Table 1. *Overview of the participants that made judgments when using Granska. The data is sorted according to the participants' activity in the study. Participant F is the only male participant in the study.*

| **Participant** | **No. of sessions with Granska that included judgments** | **No. of words in the original texts** |
|---|---|---|
| A | 13 | 14901 |
| B | 13 | 7147 |
| C | 11 | 5810 |
| D | 7 | 2193 |
| E | 5 | 1953 |
| F | 2 | 503 |
| G | 2 | 824 |
| H | 1 | 517 |
| **Total** | **54** | **33848** |

## 3.1 Results of the judgment procedure

The judgment procedure was a hard task for the users and some of them did not judge any sessions with Granska. However, six participants delivered data exactly according to the instructions, and their judgments of Granska's alarm are presented in Table 2. These participants were more active than the others and delivered more text material and made quite a few judgments. In addition to the judgments, many of the participants commented on Granska's alarms or general behavior. They made 150 comments in total. An example of one judgment and one comment is presented in Figure 2.

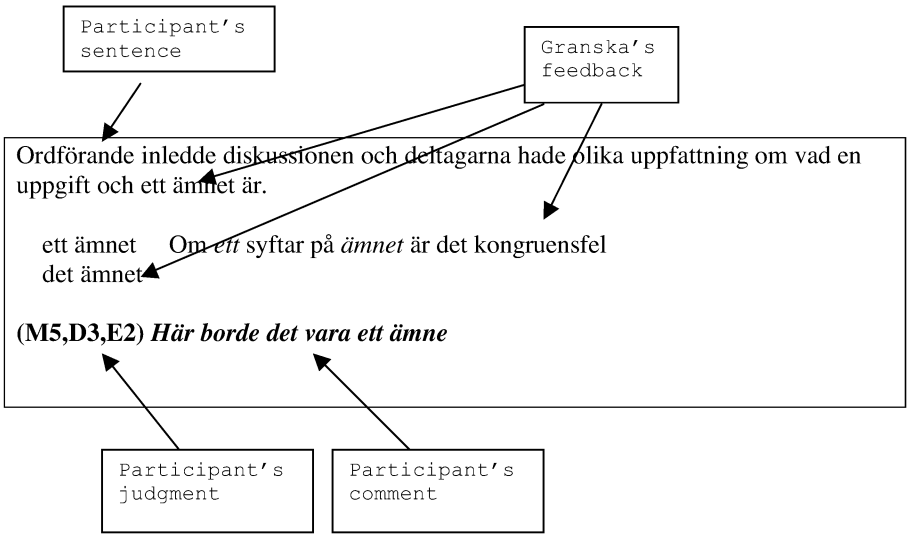


Figure 2. *Example of judgments made by participant A (written in bold within parenthesis). The letters M (detection), D (diagnosis), and E (correction) are followed by the grades. This means that the detection got a 5, the diagnosis a 3, and the correction got a 2 grade. Granska has correctly detected an agreement error (ett ämnet) with the correct diagnosis but with an erroneous proposal for correction (our analysis). Participant A made one comment in bold "**Här borde det vara ett ämne**" (eng. Here it should be: a subject). This comment points out the correct correction proposal (our analysis).*

Diagnoses and correction proposals received lower grades of the detections of the errors, see Table 2. Diagnoses and corrections include much more information about the errors than the detection does, and are therefore scrutinized more during the judgment process.

Active participants gave higher grades than participants that had used Granska only a few times.

Table 2. *Mean values from the judgments made by participants that graded each of the three components of Granska's alarm (according to the given instruction). N=number of judgments.*

| Participant | N | Detection mean | Diagnosis mean | Correction mean |
|---|---|---|---|---|
| A | 98 | 4,9 | 4,0 | 4,5 |
| B | 37 | 4,7 | 4,2 | 3,8 |
| D | 29 | 4,9 | 4,0 | 3,5 |
| E | 14 | 4,3 | 4,1 | 4,0 |
| F | 8 | 4,0 | 3,5 | 3,5 |
| H | 5 | 2,6 | 2,4 | 2,0 |
| **Total** | **191** | **4,7** | **4,0** | **4,0** |

When looking at specific error types there are three error types that received lower grades than the others (see Table 3). The error type that points out that something is missing in the clause (Missing X in Table 3 and 4) is the lowest rated one with a mean value below 3 on the diagnosis part, and even worse on the correction part with a mean of 2.0. Diagnoses and correction proposals of split compounds (Split in the tables) and disagreement between the predicative and the subject (Predicative) got grades around 3. Notable is that the judgments of the error type Spelling got lower mean values than more grammatical error types like form problems in the verb chain (Verb Chain) and disagreement within NPs (Agree NP). We will return to these issues in the discussion below.

The judgments of false alarms suggest that around half of them were not recognized as false alarms (see Table 4). This is based on the number of false alarms that got a grade lower than 3 on the diagnoses and corrections from Granska. When looking at the judgments in detail we found only 5 false detections that were judged with a grade below 3, but 11 false diagnoses and 12 false corrections, which were judged with a grade below 3. From that information and from Table 4 we can also see that the participants seemed to notice the false alarm as a false alarm when they are looking at the diagnoses and the correction proposals. Most problematic were Granska's alarms on word order problems (Word order) where all false alarms got the grade 5.

Table 3. *Judgments of all alarms that are judged individually according to detection, diagnosis and correction. N=number of judgments.*

| **Error type** | **N** | **Detection mean** | **Detection Std. dev.** | **Diagnosis mean** | **Diagnosis Std. dev.** | **Correction mean** | **Correction Std. dev.** |
|---|---|---|---|---|---|---|---|
| Agree NP | 31 | 4.9 | 0.72 | 4.7 | 0.86 | 4.6 | 1.04 |
| Mechanics | 8 | 5.0 | 0.00 | 3.8 | 1.48 | 3.0 | 1.41 |
| Misc | 4 | 5.0 | 0.00 | 4.0 | 1.73 | 4.0 | 1.73 |
| Missing X | 14 | 3.7 | 1.62 | 2.9 | 1.39 | 2.0 | 1.41 |
| Predicative | 6 | 4.7 | 0.47 | 3.7 | 1.89 | 3.7 | 1.89 |
| Spelling | 90 | 4.8 | 0.81 | 3.9 | 1.17 | 4.2 | 1.37 |
| Split | 15 | 4.1 | 1.59 | 3.5 | 1.71 | 3.1 | 1.84 |
| Verb chain | 13 | 5.0 | 0.00 | 4.7 | 1.07 | 4.7 | 1.07 |
| Word order | 10 | 5.0 | 0.00 | 5.0 | 0.00 | 5.0 | 0.00 |
| **Total** | **191** | **4.7** | **0.95** | **4.0** | **1.32** | **4.0** | **1.54** |

Table 4. *Judgments of "genuine" false alarms that are judged individually according to detection, diagnosis and correction.*

| Error type | N | Detection mean | Detection Std. dev. | Diagnosis mean | Diagnosis Std. dev. | Correction mean | Correction Std. dev. |
|---|---|---|---|---|---|---|---|
| Agree NP | 1 | 5 | 0 | 2 | 0 | 2 | 0 |
| Mechanics | 5 | 5 | 0 | 3.4 | 1.62 | 3.4 | 1.62 |
| Misc | 1 | 5 | 0 | 1 | 0 | 1 | 0 |
| Missing X | 3 | 2.3 | 1.89 | 1.7 | 0.94 | 1.7 | 0.94 |
| Predicative | 1 | 4 | 0 | 1 | 0 | 1 | 0 |
| Spelling | 8 | 4.0 | 1.73 | 3.3 | 1.64 | 2.6 | 1.58 |
| Split | 3 | 3.7 | 1.89 | 3 | 1.63 | 2 | 1.41 |
| Word order | 4 | 5 | 0 | 5 | 0 | 5 | 0 |
| **Total** | **26** | **4.2** | **1.57** | **3.1** | **1.72** | **2.8** | **1.71** |

The participants made many more comments when judging Granska's alarms than we expected; in total 150 comments. The tendency was that participants that made many judgments also wrote many comments. Many of the comments contain information about the error types that were not detected by Granska, and especially those errors that the teacher had found but Granska had not. Some comments contain the user's motivation for a certain judgment. The comments also gave us information on how the participants reflect on Granska's behavior and its feedback. A few of them used quite advanced grammatical language in their reflections on Granska's "reasoning", like for instance the term "ellipsis", while others used a language close to speech.

## 3.2 Results from observations and interviews

Tracking what happens when a false alarm occurs is complex. We compared the judged text versions and the final text versions as a means to follow the users' decisions. In our material there are cases where the users followed the false alarms, indicated by a high grade in the judged text version and a change in the final text version. There are also other cases where the false alarm got a high grade but when looking in the final version they have not followed Granska's false advice. Granska is observably not the only tool the learners use in their text revision; other resources and tools, internal as well as external, must have been used. We will return to this issue in the discussion below.

We observed that the teacher is very important as a mediator of the feedback from the program. The learners asked many questions of the teacher on problematic alarms from Granska. Naturally it was hard for the teacher to answer questions on Granska's internal reasoning. Instead the teacher focused on grammaticality, acceptability and correctness. Some of the participants thought that Granska should be an interactive environment where correction proposals could immediately be put into the original text. Some others had problems using Granska beside the word processor, and they also had problems with attaching texts to mails or saving files in different formats. Others were keen about using computers and showed no problem in shifting from one program to another using several windows, and transferring text between them.

As a part of their revision process some of the participants first used Microsoft Word's grammar checker and then Granska. The participants appreciated the meta-

language used in Granska, although they did not understand all terms. They had problems with false alarms and also choosing between different proposals for correction, and some of them used external resources for this process such as friends and dictionaries. On the other hand, other students liked the fact that one error could get several diagnoses. A few of the users did not believe that Granska could be useful for students on lower levels; to use Granska they said, one needs to be at the same level as the users in the study. Some participants appreciated the colors used in Granska for the indication of detection, diagnosis and correction proposals. Our questions about new functionality (i.e. other tools than grammar checkers) were hard for them to answer. However, they wanted Granska to find more errors, give better diagnoses, be interactive and that examples should be used in user dialogues.

# 4 Discussion

The studies helped us to gain insight into the problems that can arise when using grammar checkers as well as the kind of support second-language learners need.

## 4.1 Lack of adequate feedback and misleading feedback

The fragmentary feedback that current grammar checkers provide is a well-known problem (cf. McGee & Ericsson, 2002). Grammar checkers can give valuable feedback on some constructions, but none or only confusing feedback on other constructions similar to the ones that were detected and diagnosed. For participants, it was problematic that Granska gave feedback on some texts, and nearly none on other texts. A consequence of the latter could have been that participants thought they had written an error-free text. For the participants, the fragmentary feedback contributed to a general lack of trust in the program's possibilities. This was shown in the comments made in addition to the judgment procedure.

The interviews, and the comments made during the judgment procedure, showed that there was a lack of specific feedback regarding certain error types, and a lack of correction proposals. For example, one frequent wish was to get a better coverage of word order phenomena from the program. Granska generally does not cover word order phenomena. A comment from a participant may illustrate the need for a more comprehensive detection of such errors: "it does not matter how many grammar books I read, I will always make these word order errors".

Solecisms can occasionally be the only problem that Granska detects in a text. Some solecisms initiated discussions between the teacher and the students about the content and behavior of Granska. Other solecisms got high grades from the participants in their judgments. The solecisms steal attention from more important error types, and are a problem for both learners and the teacher. However, learners will get comments on solecisms from people around them. One lesson that we have learned about solecisms is that they should not be pointed out as errors, except on the user's demand and as what they are: comments on correctness.

The inventory of error types must be enlarged, and more qualified. But it is also important to always give the user some kind of feedback. Even though the coverage of the grammar checker can be improved, it will never be complete. We must find other ways to provide the users with relevant feedback, to increase the possibilities for reflections on how the target language works.

## 4.2 Use of different sources of linguistic information

Participants having written long texts and using Granska for revision were all misled by the program – they followed erroneous advice from the program. This happened only a few times for each participant, but helped us understand that providing the user with other sources of information than just the spelling- and grammar checker is necessary. One example that illustrates the need for different sources of linguistic information is the correction proposals from the spelling-checking module, which are sometimes hard to judge for the participants. In the participants' judgments, we can see low grades on both the diagnosis and the correction proposals. The spelling checker is often the only knowledge source available for the user (a least on the computer). In the studies, we observed that some of the participants used a dictionary when the program could not provide a correction proposal on a suspected error.

The interviews together with the judgments of Granska's alarms showed that during writing, participants were using different sources of linguistic information. For instance, some of the participants first used Word's grammar and spelling checker, and then they used Granska. They also asked friends for solutions on their language problems. Incorporating several sources of knowledge into their decision-making seems to be natural, and Granska worked as one of the sources. Participants seemed to view the teacher as the only language expert. The programs, friends or books are resources that are available when writing at home, and they were using them in parallel.

## 4.3 Focus on form

During the course, the teacher scrutinized all the students' writing assignments and corrected them by providing writing feedback on their language errors. The teacher also discussed common and significant errors. The errors cannot be neglected and one wish from the teacher in connection with the students' use of Granska, was to get rid of low-level errors. If the students' texts were free from most low-level errors, the teacher could focus on high-level errors and work with important questions in composition, like adaptation to text genre. Therefore a program like Granska, seems to be an important part of a language learning system.

Despite the important problems that a tool like Granska presented during its use in a second-language learning context, the teacher and the students welcomed Granska. That was probably due to the fact that Granska can in many cases limit the amount of repeated low-level errors.

## 4.4 Trusting the program

The teacher and the students were worried about false alarms produced by the program, and how students could deal with them. In the annotated texts, many alarms from Granska were compared with the teacher's comments and error annotations. This was mostly done in the users' initial sessions with Granska. Participants who had used Granska three or four times seemed to be aware of the pros and cons of the program, and to know how to ignore irrelevant repeated alarms. Irrelevant alarms in this context are alarms that the users did not know what to do with. Error types that were frequently ignored were *finite verb missing* and *subject missing*. Unfortunately, a majority of these alarms were relevant, but the meta-language used in the program's diagnoses was difficult to understand. We observed that without any proposal for

correction, all these alarms were neglected. On these matters, the teacher played an important role, s/he functioned as a learner's partner in the revision of the text, "translating" the grammar checker's feedback and its behavior. The teacher is essential in the relation that learners can develop with a language tool, as the teacher represents the reliable and external source of linguistic knowledge.

## 4.5 Meta-language and grammatical knowledge

When participants were asked about the diagnoses, most of them said that they appreciated grammatical terminology although they did not fully understand it. They also said that all grammatical terms should be explained within the program's help system, and the descriptions should be concise and short. These texts should also be illustrated with examples on grammatical constructions based on common words that are easy to understand.

One of the learners seemed to have extraordinarily good grammatical knowledge, which was important when judging the alarms from Granska. She could analyze her own sentences, using a proper grammatical language, and thereby reason and understand when the program was wrong. In many cases, she used only the program's detection of the error when revising her text. Another learner, with better writing skills, and more fluent Swedish, but without the first user's grammatical knowledge, deployed a similar strategy for judging Granska's alarms. From her judgments and comments, we can see that she expected more general grammatical knowledge from the program, as a complement to Granska's specific alarms. An example of this is that she wanted to know the spelling rules and not only the correct word form when interacting with the spelling checker. She wanted to learn the rules of the language, and she wanted the program to explain and provide them. That the program gave her a solution on a specific language problem was not enough for her; she obviously wanted to learn at an explicit level, not only to produce correct Swedish sentences.

Other learners did not use grammatical terminology when reasoning about Granska's alarms. They mentioned in the interviews that without correction proposals it was hard to understand the program's alarms.

To conclude, one group of well-motivated students sought for strategies to analyze the program's alarms, and to learn from that process. Another group did not have the determination to search for what the program was pointing at. For them it was more important to judge the usability of the program's alarms in accomplishing the writing task.

## 4.6 Transparency

Frequent questions posed by the students and the teacher were: Why does not the program detect this error that is so obvious? Nothing in the interface showed them how the program had analyzed the construction. The only information available was the part-of-speech tags from Granska's underlying analysis. The tags were not informative to either students or teacher. One important design issue was raised; how should we make the interface to Granska's analyses and reasoning more transparent?

From the judgments and the comments made by the users, we understood the importance of that question for the users; in other words, the importance of relating

their own reasoning about language to the program's reasoning and analysis. As we mentioned before, the teacher is an important guide in this process, and the program's knowledge should at least be accessible to the teacher.

A program in a language learning system should try to explain how it analyzes language, and on what grounds it gives feedback. Every level in the analysis and generation of language seemed to be relevant for those learners wanting to learn something from the program.

### 4.7 Interaction and integration

Many of the learners hoped that Granska would be more interactive. They started to click on the web page generated by Granska. When using Granska's web interface it was necessary to edit the text in another program, and at the same time to look at the feedback from Granska. We had to instruct them about the use of the different windows open on the screen, one for Granska, and one for their usual word processing program.

When instructed, learners thought it was double work to cut and paste the correction proposal from the web page into their texts. They also felt frustration when wanting to use Granska a second time on the revised version of the text. In our instructions to the users, we encouraged them to use an on-line dictionary (Lexin), which was accessible through a web interface. Thereby they had three different windows to handle. For the well-motivated learners having good computer literacy, the management of the windows was quite easy, but for the others, that was extra and demanding work resulting in a loss of concentration on the task at hand.

A language learning system must give interactive feedback, and the different tools should be integrated, so that the learners do not lose attention when using them in parallel. It is also preferable that the feedback is immediate, and integrated in their writing process.

## 5 Designing a language environment focused on learning Swedish

One conclusion from our studies was that a spell- and grammar checker (even if it is powerful and robust) is not enough. The learners did not only want feedback on their mistakes, they also wanted explanations going beyond the capabilities of a grammar checker (at least the standard behavior of a grammar checker), as well as more possibilities for language production. They did not want only one proposal from the spelling checker; they wanted information about e.g. other word forms from the same lemma, meta information about the word chosen, and which spelling rule they had violated. Another important insight of the studies was that a tool is not good enough on its own, but together with other tools and the teacher's guidance, it can become a useful toolkit for the user. By using different sources for language analysis and understanding, the users can more easily make their own decisions on their own language production.

To meet some of the learners' needs, we have recently developed an interactive language environment for Swedish called Grim[4]. Grim builds on Granska, but also integrates other tools into a flexible user interface. The functionality in Grim is based on independent programs running on different servers; the program on the user's computer is just a client, with basic word processing functionality, and with interactive user interfaces to the independent programs to give immediate feedback. However, for the user, Grim appears as one single program with a seamless integration of several language tools.

The design of Grim is grounded in insights gained during the user studies. In short, the user studies helped us understand that second-language learning environments should support:

- Focus on form, providing the learner with explicit lexical and grammatical feedback.
- Focus on authentic language use, providing the learner with implicit linguistic knowledge.
- Language exploration and reflection, for instance by visualizing grammatical categories and concordances.
- Different and competing views of the learner's interlanguage and the target language.

According to Chapelle (1998) it is important to support the learner's apperception and comprehension of target language input. A computer-aided language learning program can make these processes salient by highlighting important aspects in the target language, for instance grammatical categories that are important for the learner's linguistic system to develop. However, the learner's production must also be given relevant feedback, for example by making the learners aware of errors in their language production, and how to correct those. It is especially important that these processes are made salient in activities that include a communicational goal.

Five tools based on language technology and aiming to support these processes have been integrated into Grim. The tools are:

1. The grammar checker Granska, which also is the most developed tool,
2. a surface syntactic parser, called GTA, which has grown from the work with Granska,
3. a concordance interface to the Swedish version part of the Parole corpus (19 million words from novels, newspapers, and journals),
4. a dictionary with eight language pairs called Lexin containing 28 500 words and
5. an interface to a tool for automatic word inflection.

## 5.1 Supporting focus on form

According to Long, focus on form refers to how focal attentional resources are allocated. "Although there are degrees of attention, and although attention to forms and attention to meaning are not always mutually exclusive, during an otherwise

[4] Grim is freely available at www.nada.kth.se/grim. Grim is written in Java and works under most operating systems.

meaning-focused classroom lesson, focus on form often consists of an occasional shift of attention to linguistic code features – by the teacher and /or one or more students – triggered by perceived problems with comprehension or production" (Long and Robinson, 1998, p. 23). From the user studies, we observed that focus on form was important for the learners and the teacher. Furthermore findings on second-language acquisition suggest that when second language learning is entirely experiential and meaning focused, some linguistic features do not ultimately develop to target-like levels (Doughty & Williams, 1998; Harley, 1992).

When designing the interface to Granska in Grim, we focused particularly on the interaction between the user, the text and Granska. The feedback should be immediate, and the text should be easily updated with the correction proposals from Granska. Another important feature in the interface design was to put the preferences and options for the selection of error types on the surface of the user interface. As default, all errors are underlined with red. However, the user can change the color of the highlighting of every error type by using a color palette. As default, the error types are visible in the left column of Grim (see Figure 4). The error diagnoses and the correction proposals are on the demand of the user visible in the right column of Grim (see Figure 4).

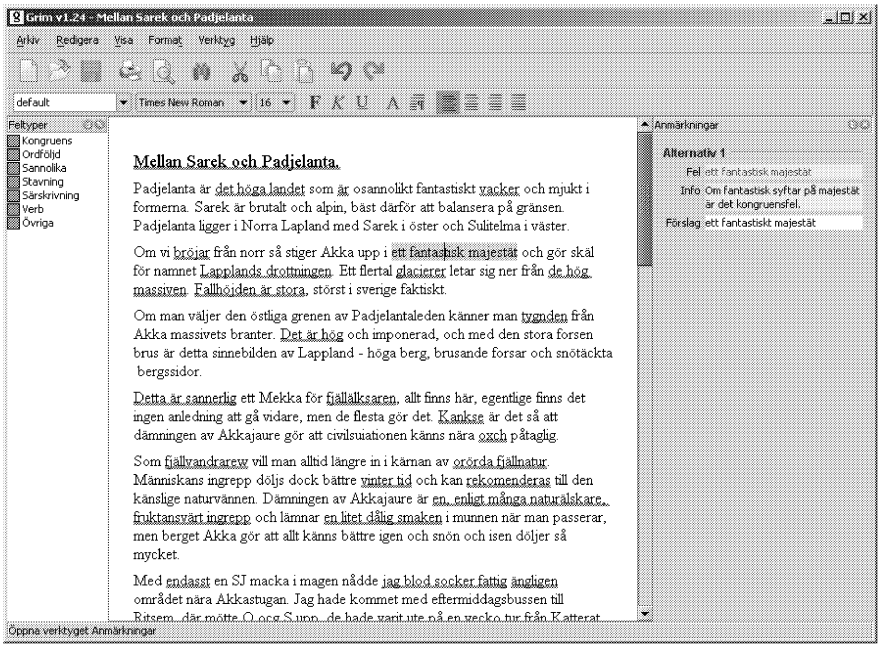


Figure 4. *Errors in a text processed in Grim, highlighted by the grammar checker Granska. The error types that Granska can detect are visible in the left column. The colors of the underlinings can be changed or made invisible. Feedback on an error is presented in the right column of Grim.*

## 5.2 Visualization of grammatical categories

Grammatical knowledge and reasoning about this knowledge are important processes during different phases of writing, and are important in the concept focus on form. In our studies, we observed that learners reasoned in order to determine and correctly use the alarms from Granska. Therefore, beside Granska in Grim, we developed a tool to visualize grammatical categories in text, based on the output from the surface oriented parser GTA (Knutsson et al, 2003). The parser is designed for robustness against ill-formed natural language data, and it has been carefully evaluated on this task (Bigert et al, 2003).

When interacting with this tool, the user can select grammatical categories that should be highlighted in their texts. Thereby, the program does not spell out the grammatical rules of Swedish, instead the application of the grammatical rules are shown to the user. As an example, all Swedish noun phrases in the text can be highlighted (according to the parser's view of Swedish), and it can be explained how words in a Swedish noun phrases are grouped together. In addition, by using this function together with Granska, the Swedish agreement system within noun phrases can be visualized. The idea can be illustrated by a citation from Warschauer and Healey (1998) "Having discovered the linguistic rules themselves, students are more likely to remember them". The highlighted words are examples of how grammatical categories and rules are connected to the user's own words. The tool can be used to facilitate implicit focus on form tasks.

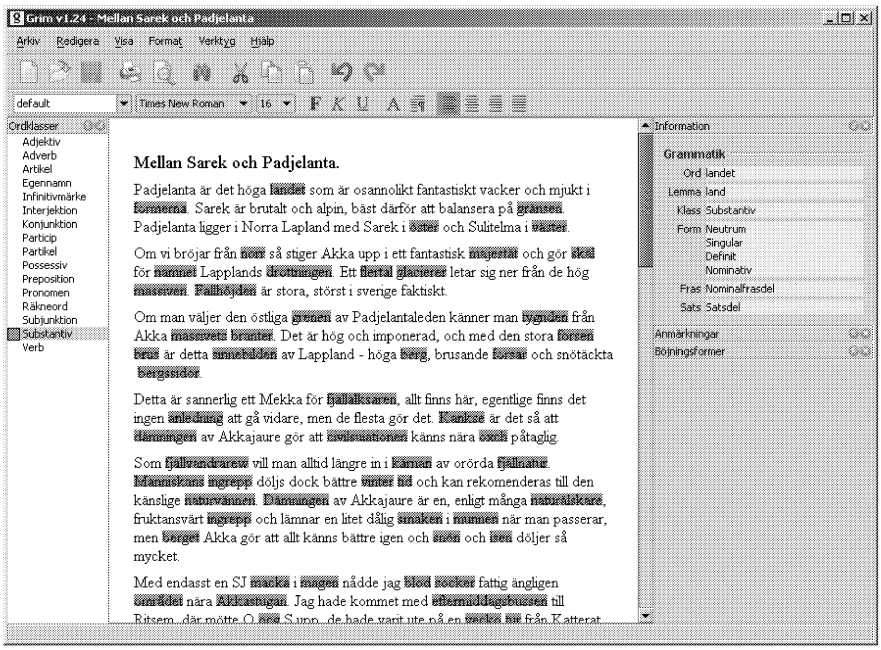


Figure 5. *Nouns are highlighted with red in Grim. 16 different word classes can be highlighted using the palette in the left column.*

The language tools implemented in the program Grim contain fine-grained information about grammatical categories and error types. There is a problem with representation here, when for instance only information about word classes and different inflections give 145 possible choices for the user. In the first phase of the development of Grim, we chose not to include all these possible types in the user interface for highlighting grammatical information in a whole text. Instead, we represented 16 basic word classes in Swedish, see Figure 5. There is also a mismatch between what the computational linguist defines as a word class and what is taught in Swedish as a second language. We have tried to adapt the terminology to what is taught in second language courses. In the right column (see Figure 5), fine-grained grammatical information can be presented on the user's demand by pointing with the mouse on a word in the text. In this way, the feedback goes two ways, one way on the demand from the user, and highlighted with colors, and the second way through the information in the information column to the right.

## 5.3 Focus on authentic language use

Swedish is an inflection language. That means that learners have to learn different inflections of nouns, adjectives, verbs and pronouns. From the studies, we observed that some of students expected more information on inflections, especially when the spelling checker proposed error correction. By clicking on a word and selecting the word inflection function, the user obtains all inflections of a word in the information column to the right (see Figure 6).

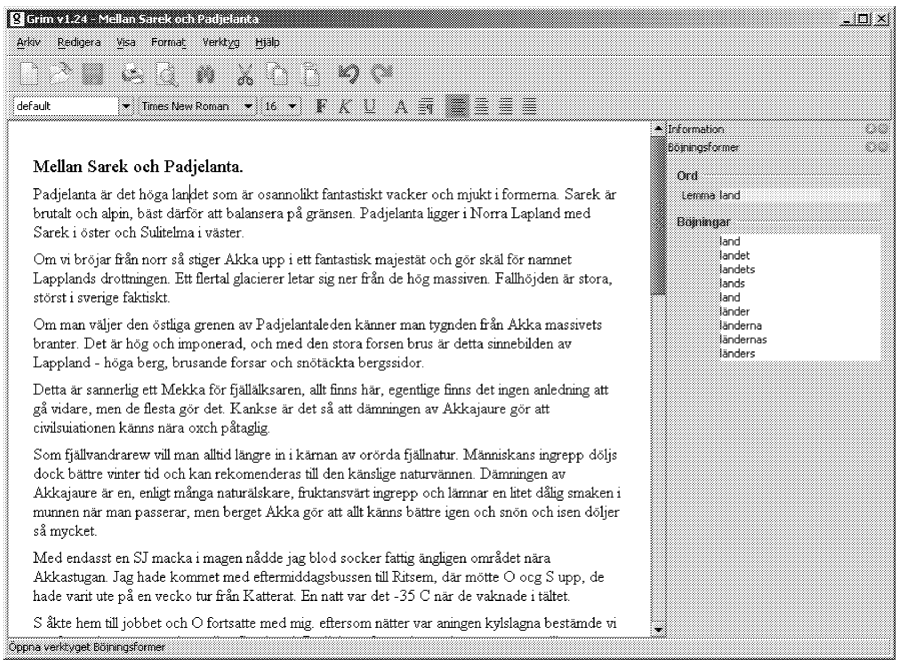


Figure 6. *Information about a word's inflection is presented in the information column in Grim. The user places the mouse pointer at a word, and all inflections of the word are presented in the right information column.*

So far, the tools in Grim have focused on grammatical feedback, but semantically oriented feedback is also important in second language acquisition (Chapelle, 1998; Salaberry, 1999). The comprehension of meaning can partly be supported by a look-up of isolated words using a dictionary. In Grim, this dictionary is called Lexin and includes eight language pairs (see Figure 7). For the users in the study present above, that is not enough because of their heterogeneous background. However, an easily accessible dictionary will hopefully encourage both the comprehension of target language input, as well as target language production.

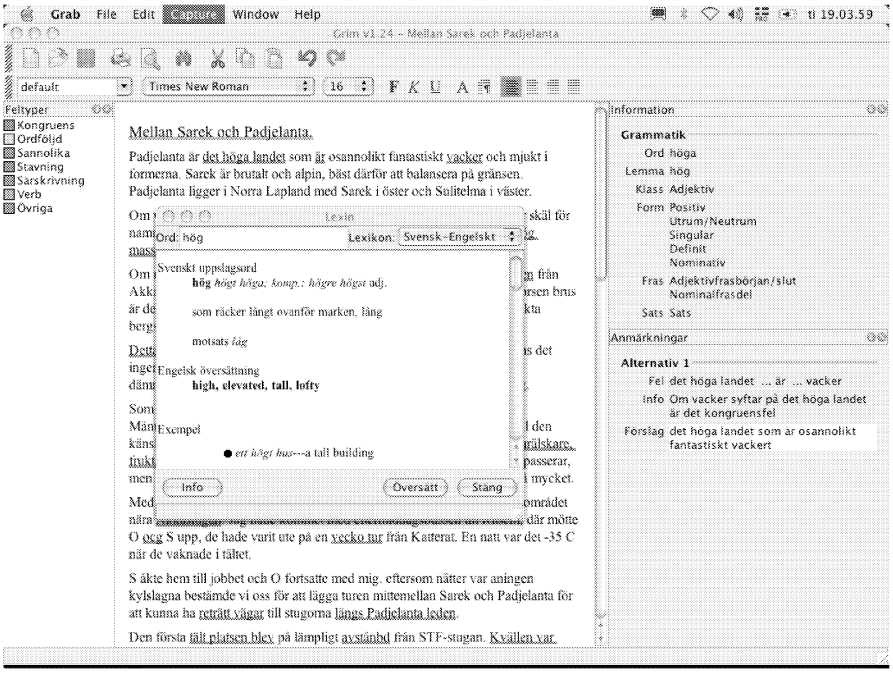


Figure 7. *Dictionary look-up of the word "hög" (eng. high) in the lexicon Lexin. This screenshot also shows how Grim looks on Mac OS X.*

## 5.4 Examples of language use

Many researchers have point out the importance of target input in second language acquisition (Breen, 1987; Candlin, 1987; Chapelle,1998; Long and Crookes, 1992). In Grim, the user can have important grammatical categories highlighted in any text. However, the user has to select and load the texts into the environment. Also, as target language input is important, and so is the quality of the learning material, to add a selection of language material seems to be an important part of a language environment. From our studies, we know that doing a corpus search using a concordance program is one lesson in the Swedish course. Therefore, it was important to add a selected language source that is searchable and can be used as an important tool for the exploration of language use in Grim.

The solution chosen was to make an interface to the Parole corpus, which contains about 19 million words (see Figure 8). The user can search for every word in her text, just by clicking on the current word and, choosing "Parole" from the tool-menu. No search expression has to be written, which is often the case in standard concordance programs. However, if the user wants to specify more details about the search, it is also possible to create more advanced search expressions.

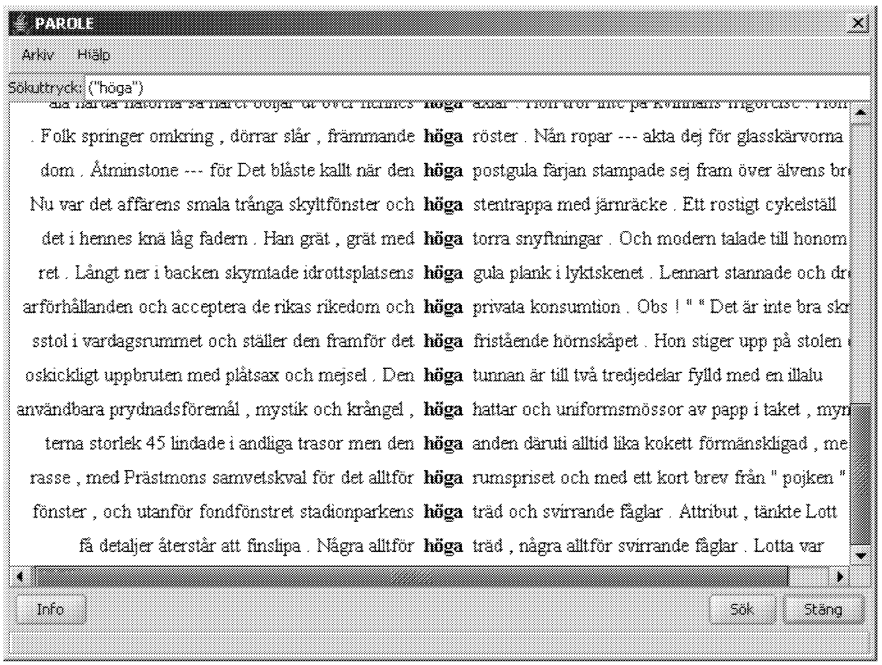


Figure 8. *The user interface to the concordance engine. Concordances for the word "höga" (eng. high) in the Parole corpus.*

# 6 Conclusions

The user studies as well as the experiences from the development and design of the language environment Grim can be summarized in the following observations.

Grim is not pedagogically neutral. The tools that are important in Grim are included in the system with the purpose to support both deductive and inductive learning. However, the learners can use them as they like, Grim does not control how, when and if the user is using them. When teachers want the learners to use Grim, they must invent their own instructions and pedagogical settings for the usage of the program. Second language learning and pedagogy are fields with no straight answers to questions related to the most effective way to teach and study a second language. Evaluation of learning effects of a specific pedagogy, or even a computer program, is a difficult task; the variables and factors are many. A current learning activity cannot be isolated from other activities that do not necessarily focus on learning, but have

learning effects. Even though not all teachers embrace language-learning environments, the learners will probably use them, if they are available. And who will stop the learners from using tools that they believe will help them when using the second language?

In the field study, we focused on second-language learners' needs and writing activity and therefore not on the teacher's role. However, we observed that the teacher played an essential part in the relation that learners can establish and develop with the language program; when introducing the tool, and subsequently when the learners started to use and learn how to use the tool. The teacher's role in language learning systems has to be further explored. When the systems become more complex, and include more language technology, they become more difficult to understand. In the teaching situation, it is natural that the teacher can explain and recognize the system's behavior.

From the developer's perspective, missing alarms are a problem of nearly the same dignity as false alarms. The decision between flagging, and not flagging an error, is one of the major internal design issues. But the teacher and the learners in the studies do not seem to have worried about this problem, they seem to be certain that a grammar checking program will find some errors in the learners' texts, and that the teacher will find the rest. The major trouble for them was the false alarms. As Vernon (2000) suggested, it is important to learn about the capability and limitations of a grammar-checking program. In our studies we observed that learners learned about the tool by comparing the teacher's comments with the program's alarms. All language learning environments, including language technology, will have problems with the program's accuracy. A crucial issue is how to design a system which enables the user to understand its limitations.

## Acknowledgements

We want to thank the students and the teacher in our field study. We are also grateful to different users from the outside world that are using Grim, and suggesting valuable comments on the program. We are thankful to Jonas Sjöbergh, Johnny Bigert, Johan Carlberger and Viggo Kann at KTH Nada for the development and maintenance of the servers running important language tools in Grim. We are indebted to Språkdata in Gothenburg for letting us connect to the concordance engine. The work has been funded by the Swedish Research Council (VR).

## References

Bender, E. M., Flickinger, D., Oepen, S., Walsh, A., & Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in CALL. *Proceedings of the InSTIL/ICAL Symposium: NLP and Speech Technologies in Advance Language Learning Systems.*

Bigert, J., Knutsson, O. & Sjöbergh, J. (2003). Automatic evaluation of robustness and degradation in tagging and parsing. *Proceedings of the International Conference on Recent Advances Natural Language Processing*, 51-57.

Bigert, J., Kann, V., Knutsson, O. & Sjöbergh, J. (2004). Grammar checking for Swedish second language learners. In P. J. Henrichsen, *CALL for the Nordic Languages*, Copenhagen, Denmark: Samfundslitteratur.

Birn, J. (2000). Detecting grammar errors with Lingsoft's Swedish grammar checker. In T. Nordgård, *Proceedings of 12th Nordic Conference on Computational Linguistics*, 28-40.

Bolt, P. & Yazdani, M. (1998). The evolution of a grammar-checking program: LINGER to ISCA. *Computer Assisted Language Learning*, 11(1).

Breen, M. (1987). Learner contributions to task design. In C. Candlin, & D. Murphy, *Language learning tasks,* Lancaster practical papers in English language education, Vol. 7, Englewood Cliffs, NJ: Prentice Hall.

Candlin, C. (1987). Towards task-based language learning. In C. Candlin, & D. Murphy, *Language learning tasks*, Lancaster practical papers in English language education, Vol. 7, Englewood Cliffs, NJ: Prentice Hall.

Cerratto, T. (1999). Activite´ collaborative sur re´seau. Une approche instrumentale de l'e´criture en collaboration. (Collaborative networked activities. An instrumental approach to collaborative writing) Ph.D. thesis. University of Paris VIII- St. Denis.

Chapelle, C. A. (1998). Multimedia CALL: Lessons to be learned from research on instructed SLA. *Language learning & Technology*. 2(1), 22-34.

Chapelle, C. A. (2001). *Computer applications in second language acquisition. Foundations for teaching, testing and research*. Cambridge: Cambridge University Press.

Cornu, E., Kübler, N., Bodmer, F., Grosjean, F., Grosjean, L., Léwy, N., Tschichold, C. & Tschumi, C. (1996). Prototype of a second language writing tool for French speakers writing in English. *Natural Language Engineering* 2 (3): 211–228.

Domeij, R., Knutsson, O., Carlberger, C & Kann, V. (2000). Granska – an efficient hybrid system for Swedish grammar checking. In *Proceedings of 12th Nordic Conference on Computational Linguistics*, 49-56.

Domeij, R., Knutsson, O. & Severinson Eklundh, K. (2002) Different ways of evaluating a Swedish grammar checker. *Proceedings of The Third International Conference on Language Resources and Evaluation (LREC 2002)*, 262-267.

Doughty, C. & Williams, J. (1998). *Focus on form in classroom second-language acquisition*. Cambridge: Cambridge Applied Linguistics.

Fathman, A.K. & Whalley, E. (1990). Teacher response to student writing: Focus on form versus content. In B. Kroll, *Second Language Writing*. Cambridge: Cambridge University Press.

Ferris, D. R. (2004). The “grammar correction” debate in L2 writing: Where are we, and where do we go from here? (and what do we do in the meantime…?). *Journal of Second Language Writing*, 13, 49-62.

Garrett, N. (1991). Technology in the service of language learning: Trends and issues. *Modern Language Journal*, 75, 74-101.

Gass, S. M. (1994). The reliability in L2 grammaticality judgments. In E. E. Tarone, S. M. Gass & A. D. Cohen, *Research methodology in SLA*. Hillsdale, NJ: Lawrence Erlbaum, 303-322.

Goss, N., Z. Ying-Hua & J. P. Lantolf. (1994). Two heads may be better than one: mental activity in L2 grammaticality judgments. In E. E. Tarone, S. M. Gass & A. D. Cohen, *Research methodology in SLA*. Hillsdale, NJ: Lawrence Erlbaumm, 263-286.

Harley, B. (1992). Patterns of second-language development in French immersion. *Journal of French Language studies,* 2(2), 159-183.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T & Isahara, H. (2003). Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL '03)*, 145-148.

James, C. (1998). *Errors in Language Learning and Use: Exploring Error Analysis.* London: Longman.

Knutsson, O., Cerratto Pargman, T. and Severinson Eklundh, K. (2002). Computer support for second language learners’ free text production – Initial studies. In M. Valcke, & A. Bruce, *European Journal of Open and Distance Learning (EURODL).*

Knutsson, O., Cerratto Pargman, T. and Severinson Eklundh, K. (2003). Transforming grammar checking technology into a learning environment for second language writing. *Proceedings of the HLT/NAACL 2003 workshop: Building Educational Applications Using NLP*, 38-45.

Knutsson, O., Bigert, J. & Kann, V. (2003). A robust shallow parser for Swedish. *Proceedings of 14th Nordic Conference on Computational Linguistics.*

Lantolf, J.P. (2000). Introducing sociocultural theory. In J.P. Lantolf, *Sociocultural Theory and Second Language Learning*, Oxford: Oxford University Press.

Laurillard, D. & Marullo, G. (1993). Computer-based approaches to second language learning. In P. Scrimshaw, *Language, classrooms and computers*. London: Routledge.

Levy, M. (1997). *Computer-Assisted Language Learning: Context and Conceptualization.* New York: Oxford University Press.

Lindberg, J. & Eriksson, G. (2004). CrossCheck-korpusen – en elektronisk svensk inlärarkorpus. In *Proceedings of ASLA-konferensen 2004*. Södertörn University College, Sweden.

Long, M. & Crookes, H. (1992). Three approaches to task-based syllabus design, *TESOL Quarterly*, 26(1), 27-56.

Long, M. & Robinson, P. (1998). Focus on form: Theory, research, and practice. In C. Doughty & J. Williams, *Focus on Form in Classroom Second Language Acquisition*, 15-41. New York: Cambridge University Press.

McGee, T. & Ericsson, P. (2002). The politics of the program: MS WORD as the invisible grammarian, *Computers and Composition*, 19(4), 453-470.

Menzel, W. & Schroeder, I. (1999). Error diagnosis for language learning systems. *Special edition of the ReCALL journal*, 20–30.

Nerbonne, J. (2002) Natural language processing in computer-assisted language learning. In R. Mitkov, *Handbook of Computational Linguistics*, Oxford University Press, 670-698.

Olson, D. (1995). Writing and the mind. In J. Wertsch, P. Del Rio, & A. Alvarez, *Sociocultural Studies of Mind.* Cambridge University Press, Cambridge. 95-123.

Park, J. C., Palmer, M., & Washburn, G. (1997). An English grammar checker as a writing aid for students of English as a second language. In *Proceedings of 5th Conference on Applied Natural Language Processing*.

Salaberry, R. (1999). CALL in the year 2000: Still developing the research agenda. *Language Learning and Technology*, 3(1), 104-107.

Schneider, D. & McCoy, K. (1998). Recognizing syntactic errors in the writing of second language learners. In *Proceedings of Coling-ACL'98*, 1198–1204, Montreal.

Scott, M. (2001). Written English, Word processors and Meaning Making. A semiotic perspective on the development of adult students' academic writing. In L. Tolchinsky, *Developmental Aspects in learning to write*. Dordrecht: Kluwer.

Swain, M. (2000). The output hypothesis and beyond: Mediating acquisition through collaborative dialogue. In J. P. Lantolf, *Sociocultural Theory and Second Language Learning*, Oxford: Oxford University Press.

Säljö, R. (1996). Mental and physical artifacts in cognitive practices. In P. Reimann & H. Spada, *Learning in humans and machines. Towards an interdisciplinary learning scienc*e. London: Pergamon.

Truscott, J. (1999). The case for ''the case for grammar correction in L2 writing classes'': A response to Ferris. *Journal of Second Language Writing*, 8, 111–122.

Vandeventer Faltin, A. (2003). Syntactic error diagnosis in the context of computer assisted language learning. Ph.D. thesis, University of Geneva.

Verillon, P. & Rabardel, P. (1995). 'Cognition and artefacts : A contribution to the study of thought in relation to instrumented activity'. *European Journal of Psychology of Education*, Vol. X:77-103.

Vernon, A. (2000). Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities*, Computers and Composition***,** volume 17(3), 329-49.

Vygotsky, L. (1962). *Thought and language.* Cambridge, MA: MIT Press.

Vygotsky, L. (1978). *Mind and Society: The development of higher psychological processes*. In M. Cole, J. Steiner, S. Scribner, S. & E. Souberman, Cambridge, MA: Harvard University Press

Warschauer, M. (1996). Computer-assisted language learning: An introduction. In S. Fotos, *Multimedia language teaching*, 3-20. Tokyo, Japan: Logos International.

Warschauer, M., & Healey, D. (1998). Computers and language learning: An overview. *Language Teaching*, 31, 57-71.

Warschauer, M. & Meskill, C. (2000). Technology and second language learning. In J. Rosenthal, *Handbook of undergraduate second language education*. New Jersey: Lawrence Erlbaum.

## Vitae

Ola Knutsson, Computational Linguist, Ph.D. student in Human-Computer Interaction.

Teresa Cerratto Pargman, Ph.D. in Cognitive Psychology, Lecturer in Human-Computer Interaction.

Kerstin Severinson Eklundh, Ph.D. in Communication Studies, Professor in Human-Computer Interaction.

Stefan Westlund, Research Engineer and Programmer.