DEVELOPING A GRAMMAR CHECKER FOR SWEDISH[1]

Antti Arppe
Lingsoft, Inc. / University of Helsinki
antti.arppe@iki.fi

A grammar checker for Swedish, launched on the market as Grammatifix, has been developed at Lingsoft in 1997-1999. This paper gives first a brief background of grammar checking projects for the Nordic languages, with an emphasis on Swedish. Then, the concept and definition of a grammar checker in general is discussed, followed by an overview of the starting points and limitations that Lingsoft had in setting up the Grammatifix development project. After this, the initial product development process is described, leading to an overview of the error types covered presently by Grammatifix. The error treatment scheme in Grammatifix is presented, with a focus on its relationship with the error detection rules. Finally, the error types included in Grammatifix are compared to those of two other known projects, namely SCARRIE and Granska.

1. Introduction

Software programs designated as grammar checkers have been developed since the 1980's, first and foremost for English, but also for other major European languages (Bustamante & Léon 1996). Similar endeavors for the Nordic languages have been scarce, the notable exception being the Virkku system for Finnish. Virkku was developed and launched on the market in 1991 by Kielikone Ltd <http://www.kielikone.fi> as a side-kick of the company's long-term efforts in developing a machine translation system from Finnish to English. Despite this technical background, Virkku does not use the full-scale deep-syntactic parser developed for Kielikone's machine translation system, but is instead based on a lighter, unification-based approach.[2] Unfortunately, the Virkku system remains publicly undocumented.

In the case of Swedish, some level of checking of noun phrase internal agreement, based on shallow parsing, was incorporated into the Swedish version of the former Inso's International ProofReader proofing tools software, developed in cooperation with IBM in the early 1990's.[3] Nevertheless, it was not until the middle 1990's that several independent projects were initiated, more or less within the same timeframe, with the intent of developing a full-fledged grammar checker for Swedish, namely Granska, SCARRIE, and Grammatifix. The Granska project <http://www.nada.kth.se/theory/projects/granska/> was originally initiated in 1994 at the Department of Numerical Analysis and Computer Science (NADA) at the Royal Institute of Technology (KTH) in Stockholm, and has been continued on several occasions (Domeij et al 1996, 1998). The SCARRIE project <http://www.scarrie.com>, which in addition to Swedish also aimed at covering the two other main written Scandinavian languages, Danish, and Norwegian Bokmål, was started in 1996, and was scheduled to end in 1999. In the SCARRIE project, the main responsibility for the Swedish component was undertaken by the Department of Linguistics at the University of Uppsala (Sågvall Hein 1998). Grammatifix is the result of a product development project initiated in 1997 and completed in 1999 at Lingsoft, Inc., a Finnish language engineering company <http://www.lingsoft.fi>. Lingsoft has licensed Grammatifix to Microsoft as the grammar checking component of the Swedish version of Microsoft Office 2000, launched on the market in the year 2000, and has also released Grammatifix on the Swedish market as a stand-alone product under the Grammatifix brand name. Actually, there is a fourth Swedish proofing tool on the market that covers some error types traditionally associated with grammar checkers, namely Norstedts'

Skribent <http://www.norstedts.se>, but since it does not include any syntactic error detection, it was left outside the scope of this paper.

This paper outlines the development process of Grammatifix undertaken at Lingsoft. The emphasis of this paper is on general product definition and product development issues associated with such linguistic tools as a grammar checker, whereas the actual mechanism for detecting Swedish grammar errors and its linguistic principles are covered in a separate paper by Birn in the same volume. Furthermore, this paper gives an overview of the features of Grammatifix, and compares these with the other known and documented Swedish grammar checkers, namely SCARRIE and Granska.

2. What is a grammar checker – really?

In developing a grammar checker for any language, the first issue to be tackled is what type of a proofing tool is indeed going to be developed. Firstly, one must choose what types of linguistic features are going to be included in the tool. Secondly, one must design the functionality of the tool and its interaction with the user and with other software applications.

Concerning the linguistic features, the general notion is that grammar checkers, by virtue of their name, attempt to locate syntactic errors.[4] Though it may some day be possible with the development of our knowledge of linguistic structure and consequent computerized models, present grammar checkers do not and cannot check or validate the overall linguistic correctness of text, or syntactic for that matter. In practice, grammar checkers are limited to checking only a small subset of all possible syntactic structures. The first and obvious criterion on what these structures are depends on the syntactic character of the language, i.e. what types of syntactic interdependencies and consequent syntactic "rules" exist in the language. Thus, syntactic interdependencies which exist and can be analyzed in one language, such as subject-verb agreement in English, are, at least as far as concerns grammar checking, irrelevant in other languages that lack such a dependency, for instance Swedish, where noun phrase internal agreement is much more central as a syntactic feature.

A second but no lesser limitation on the structures that a grammar checker can attempt to cover are the linguistic formalisms available for the analysis and syntactic error detection of the language. It should be quite obvious that only such linguistic features that can be described and analyzed efficiently and broadly with existing linguistic formalisms and their technical implementations are worth spending limited development effort on. Even here, the choice of the type of computational linguistic analysis strategies, such as between rule-based versus statistical methods, or various combinations of these or other strategies, can produce varying results in different linguistic error categories. Finally, it must be noted that a grammar checker can presently only judge syntactic correctness or incorrectness. As long as a sentence or phrase is syntacticly well constructed, a grammar checker does not possess the capacity to assess the truthfulness of the utterance, especially so in the case of unrestricted, general language.

There is somewhat of a confusion or at least vagueness in the general consciousness of what grammar checkers are as proofing tools. Grammar checkers are often not, despite their name, only limited to purely grammatical or, to be specific again, syntactic features. In addition to these errors, grammar checkers typically address violations of or non-conformances with established conventions in punctuation, word capitalization, and number and date formatting. Furthermore, word-specific stylistic assessments are often included in grammar checkers. There is a historical reason for these non-syntactic errors to be included in grammar checkers, which is a result of the development of word processing software within the last decade or so, and how linguistic support features were integrated into these applications. The first practical proofing tools to come on the market were hyphenators and spell checkers, and their client applications were designed to interact with these tools on a single word basis, i.e. with one word interpreted as a string of characters between two white-space characters. Thus, a spell checker would not receive any information about the context of the word which it was checking, even though such information would sometimes have been necessary to make the correct decisions, for instance in the case of capitalization of a word at the beginning of the sentence. The practical solution for resolving such orthographical issues has been to move them up to grammar checkers, to be developed later. Consequently, at least in the parlance of international software companies, the difference between a grammar checker and spell checker is that whereas a spell checker is limited to verifying the correctness of a single string of characters between two white-space characters, a grammar checker is able to take into account longer sequences of such strings, typically sentences or paragraphs (cf. Sågvall Hein 1998). Thus, a string may be accepted by a spell checker but identified as erroneous in its context by a grammar checker.

Finally, one could very well ask whether such a dichotomy into grammar and spell checkers indeed is any longer necessary. At least in principle one could fully integrate the functionality of a traditional spell checker, i.e. orthographical verification, within a grammar checking tool, and this is most probably the direction into which the language industry is heading. The practical obstacle here, at least in the case of the proofing tools integrated within internationally available word processors, such as Microsoft's Word, is that different proofing tool components for a particular language have been licensed from different suppliers at different times, and can in such a case, of course, not be fully integrated in a straight-forward manner.

3. Lingsoft-specific starting points and limitations in the development process

Thus, there is, at least in principle, quite some level of freedom of choice or alternatives in defining and developing a grammar checker. On the other hand, it seems that the tradition of mopping all types of non-syntactic verifications which a spell checker cannot reliably cover under the umbrella of grammar checking is a self-reinforcing process – one only has to take a look at the sortiment of error types included in the three tools covered in this paper. Nevertheless, the general nature and goals of the organization undertaking a project also has an effect on the end product and project definition. For Lingsoft, being a commercial company, there were three fundamental starting points.

Firstly, the ultimate purpose of the project was to develop a finished and functioning software product that could be either licensed as such to third party organizations or sold as a stand-alone product directly on the market – a prototype would not suffice. This meant that the software had to be both designed and fully implemented to function properly and consistently, without crashing, halting or falling into a loop, not only with the well-formed demonstration cases but in any – reasonably foreseeable – situation, such as with unexpected combinations of user commands or client application function calls, or with unexpected input. To guarantee this, a systematic, and consequently tedious, specifically functional testing procedure, including the compilation of extensive testing material for this purpose had to be set up alongside the testing of the linguistic error detection rules (cf. Birn in this volume). Furthermore, the goal was to develop the end-product within a preset timeframe, which required the prioritization in the implementation of possible error types.

Secondly, it seemed the obvious choice to base the detection of grammar errors on the Constraint Grammar technology in general and its Swedish implementation, Swedish Constraint Grammar (SWECG) (Birn 1998), and benefit from the accompanying linguistic know-how. SWECG had been developed in-house as a part of the company's basic technology portfolio for some time, but had not yet been financially exploited on a larger scale. In the end, one should never underestimate the value of tested technology, even though some doubts lingered in the beginning on how successfully a formalism (or components of it) and accompanying tacit knowledge that had mainly been used primarily for descriptive morphological analysis, disambiguation and shallow syntactic analysis of *a priori* well-formed sentences could be adapted towards the normative ends of discovering badly-formed constructions.

Thirdly, the market situation on the Swedish software market in the end of the 1990's, with Microsoft Word as the dominant leader in the field of word processing, and the possibility of using Microsoft's at that time publicly available Common Grammar 1.x API (referred hereafter MS-CGAPI), led Lingsoft to choose to integrate Lingsoft's Swedish grammar checking tool directly with this word processor – an indirect form of interaction between the grammar checker and end-user. With direct integration to MS Word with MS-CGAPI, Lingsoft did not have to allocate (always) scant resources into creating an independent user interface for the grammar checker, though on the other hand we would have to adapt the general functional feature selection of the grammar checker to those that were indeed supported by the API. These functions were actually those functions that were supported in the implementation of the MS-CGAPI in the software code of the client applications that use MS-CGAPI, i.e. Microsoft Word.

A crucial, though not directly obvious consequence of this choice was that traditional spelling errors as described above would not fall under the scope of this grammar checking project. In this aspect it differs from both SCARRIE and Granska. On the other hand, Lingsoft had already developed a spell checker for Swedish which had been licensed to Microsoft and integrated in Microsoft Office 97 Service Release 1 (SR1) and subsequent versions of this product. Thus, in all phases of product development, the product development team could readily observe the interaction of the existing spell checker and the grammar checker under development in the actual environment in which they were eventually going to be used. Furthermore, since MS-CGAPI is interactive both in principle and in practice – contrary at least to the original specifications of e.g. Granska where proofing of text had originally been planned to be done in batch mode (Domeij et al 1996:2)[5] – the design of the discourse and interaction of Grammatifix through MS-CGAPI and Microsoft Word with the end-user would have to be take this interactivity into account from the very beginning. In addition, interactivity set minimum demands on the program's speed.

4. How were the features of the grammar checker eventually defined

The development of Grammatifix was originally started out as an exploratory project. At the very beginning, existing grammar checkers for other languages were investigated, both for the linguistic features that they covered and how well they performed their tasks, an activity that seems to have been undertaken by other projects (e.g. SCARRIE)[6]. After this, a general classification of linguistic error types, writing style violations and non-recommended word usage that were judged worth finding was compiled, using the linguistic intuition or personal observations of project members[7] and generally acknowledged guide and reference books of Swedish grammar and writing conventions. All reference works consulted at this phase were of Sweden-Swedish (i.e. *"riksvensk"*) origin. From the very beginning, Swedish material that the company or individual project members had access to, ranging from personal observations of errors in newspapers to actual corpora of Swedish texts at the company's disposal, was used to support this classification work by providing a source of genuine evidence for the existence and character of hypothesized error types, and for the discovery of new ones. These genuine examples would grow to form the kernel of the error corpus later used in the development and testing of the linguistic error detection rules (cf. Birn in this volume). After this stage, each error type in this classification was evaluated along two criteria. Firstly, the existence of a Lingsoft-proprietary technology (e.g. SWECG) or a public one (such as regular expression matching techniques), or any known technology or technique for that matter that could be used to detect the particular error type was assessed. Secondly, the perceived benefit and consequent priority of detecting a particular error type was evaluated.

Based on this preliminary work, a subset of error types was chosen to be pursued in earnest as a part of the actual product development project, and indeed this subset remained more or less the same until completion. However, a back door was left open to add new error types later, if a clear need would arise. The criteria for the selection of the error types were manifold. Firstly, detection of error types should be performed by or based on existing Lingsoft technology, or with a public technique available to Lingsoft. This was in practice a repetition of the previous evaluation of error types, but the underlying motivations were different. In the original classification we wanted to create a broad picture of what we and others could conceive of in a Swedish grammar checker, so that we could later see in the right perspective the set of error types we could actually cover. [8] Secondly, the errors should be truly relevant for Swedish and not merely be localizations of foreign grammar error types. Last but not least, the probability of success in discovering errors as perceived by the development group by using the chosen technologies should be judged high at the very beginning of the development process, so that the most could be made with the existing (personnel) resources within the preset timeframe, leading to the choice of error types evident with close contexts, i.e. adjacent or nearly adjacent words. From experience with SWECG it was known that the Constraint Grammar formalism showed best results in close interdependencies, and furthermore Swedish as a language exhibits a high amount of word interdependency in close contexts. As an arbitrary working goal a precision of over 67 percent for each error type was chosen, i.e. two-thirds of flaggings for each error type should be justified in order for the error type to be included in the final product. This general aim at high precision – for a grammar checker – was in line with Bernth's observations on end-user valuations, in which satisfaction was specified as high precision, i.e. few false recalls, even at a noticeable loss of recall. Even though users expect a proofing tool to find as many errors as possible, they prefer easing up on this expectation if the proportion of correct error flaggings is relatively high (Bernth 1997).

The list of the error types addressed by Grammatifix should consequently be of no great surprise, and is rather similar to those of the other projects, which can naturally be attributed to the language in question. Thus, checks on noun phrase internal agreement and verb chain consistency have a central place in the error type portfolio. All in all, Grammatifix covers 43 error type checks, of which 26 are syntactic in nature (of which 17 belong either to the noun phrase internal or to verb chain consistency error types), 14 address punctuation, number and date formatting conventions, and 3 cover word-specific non-standard stylistic usage. A more specific listing of these error types with example errors is given in Table 1 (syntactic errors) and Table 2 (non-syntactic errors). Since Grammatifix is under constant development, an up-to-date version of its error types is available on the Internet (Arppe et al 1999).

Different techniques were selected for detecting various error types. The Constraint Grammar formalism is used for the detection of syntactic errors, and this is described in depth in a separate paper by Birn in this volume. Regular expression based techniques are used for the detection of punctuation and number formatting convention violations. Word-specific stylistic marking is covered by style-tagging individual lexeme entries in the underlying Swedish two-level lexicon (SWETWOL: Karlsson 1992), which was revised and augmented in this respect for the purpose of this project. It must be noted here that even though these three different techniques form the linguistic core of Grammatifix, a substantial amount of programming work was needed to adapt and combine them into a single, consistently functioning software entity.

These error types in general seem to reflect the influence of the use of word processors in the writing process (Severinsson Eklundh 1993). In the case of syntactic errors it has been observed that, contrary to common assumptions, also mother tongue writers of a language have agreement errors in their texts. Example studies on this exist at least for Spanish (Bustamante & Léon 1996) and Swedish (Domeij et al 1996: 6). These types of syntactic errors have been explained as a result of the ease of editability of text using copy-paste techniques in word-processors, and sloppy manual proof-reading of the resultant text. Even more can syntactic errors be expected in texts written by non-mother-tongue writers of a language, of which, in the case of Swedish, there are substantial numbers in both Sweden, as a result of long-term immigration, and in Finland, due to the official bilingual status of the country. A more traditional source of agreement errors is probably still to some extent words of foreign origin, where English has become the dominant source in the last decades. Increasingly international contacts through the Internet and otherwise can also be seen as a source of potential errors, since orthographical and formatting conventions vary from language to language. Here the influence of English is, of course, again obvious. As far as concerns non-syntactic errors in general, these can for the most part be attributed to the same reasons as the syntactic ones: non-linear text production without careful, if any, scrutiny afterwards.

Table 1: Syntactic error types in Grammatifix (Swedish translations of error types in parentheses; words or segments involved in the error underlined in the examples)

1. Definiteness form of noun (Bestämdhetsform hos substantiv)	Det är i samhällets utvecklingen bort från detta som Arbetsdomstolen inte hängt med.
2. Definiteness form of adjective (Bestämdhetsform hos adjektiv)	Barnen får använda sin egna energi.
3. Number agreement: determiner and noun (Numeruskongruens: determinerare och substantiv)	I protest mot de statliga monopolet började han sälja sprit på Drottninggatan i Stockholm.
4. Number agreement: adjective and noun (Numeruskongruens: adjektiv och substantiv)	Hur skapa en synliga hand som återigen är jämbördig med den osynliga?
5. Gender agreement: determiner and noun (Genuskongruens: determinerare och substantiv)	I maj i fjol genomgick Brolin ytterligare ett operation.
6. Gender agreement: adjective and noun (Genuskongruens: adjektiv och substantiv)	Detta är alltid ett nytt regims ödesfråga [NB. ‘ett’ is marked separately as erroneous under error type 5].
7. Masculine form of adjective (Maskulinform hos adjektiv)	Då frestade han ditt kött och sände dig den rödhårige kvinnan.
8. Gender agreement: pronoun and noun (Genuskongruens: pronomen och substantiv)	Vattenfall har hittills lagt gasturbinen i Arendal i malpåse och vill sälja en av de tre aggregaten i Trollhättan.
9. Subject complement agreement (Predikativkongruens)	Då hade läget i byn redan blivit outhärdlig för gruppen.
10. Supine without the "ha" auxiliary verb (Supinum utan "ha")	De kunde fått bilderna på begravningsgästerna från danska polisen.
11. Double supine (Dubbelt supinum)	Vi hade velat sett en större anslutningstakt, säger Dennis.
12. Double passive (Dubbelt passiv)	Saken har försökts att tystas ner.
13. S-passive after certain verbs (S-passiv efter vissa verb)	Huset ämnar byggas.
14. Infinitive after preposition (Infinitiv efter preposition)	Vidare ska pengar omfördelas till bland annat satsningar på Internet för stödja myndigheters och företags miljöarbete.
15. Infinitive without an expected "att" [after a verb] (Infinitiv utan "att")	Han kunde inte undvika möta hennes blick.
16. Infinitive with unexpected "att" (Infinitiv med "att")	Axelstöd och gymnastik är bästa motmedlen om man inte vill att ha förändringar i nacken och käken som gör spelet stelt.
17. Number of finite verbs (Antalet finita verb)	I Ryssland är betalar nästan ingen någon skatt.
18. No verb (Inget verb)	Ingenting här.
19. No finite verb (Inget finit verb)	Hon börja spela cello.
20. Position of adverb in subordinate clauses (Placering av adverb i bisats)	Den har setts av så få personer på biograferna att den lär knappast gå över den magiska miljongränsen.
21. Position of negated element in subordinate clauses (Placering av negerat led i bisats)	En del håller på den gamla goda tiden och påstår att lite stryk gör ingen skada. [... inte gör någon skada.]
22. Constituent order in subordinate interrogative clauses (Ledföljd i indirekt frågesats)	Jag undrar vad gör de unga männen i Finland.
23. Double negation (Dubbel negation)	Det kan bli svårt att få jobb och om man inte har varken pengar eller familj att stöda en.
24. Use of preposition with two-part conjunctions (Prepositionsbruk vid tvåledad konjunktion)	Det är utbildning som idag inte erbjuds vare sig i Lund eller Malmö. [... vare sig i Lund eller i Malmö.]
25. Form of pronoun after preposition (Pronomenets form efter preposition)	Vi sjöng för de.
26. The construction "möjligast" + adjective (Konstruktionen "möjligast" + adjektiv)	Han körde med möjligast stora snabbhet.

Table 2: Non-syntactic error types in Grammatifix (Swedish translations of error types in parentheses; words or segments involved in the error underlined in the examples)

27. Quotation marks (Citattecken)	Vi tror att det är ”möjligt att klara detta.
28. Date expressions (Datumuttryck)	Stockholm 1998.5.19
29. Several spaces in a row (Flera mellanslag i rad)	Det största är_arbetslösheten.
30. Multipart abbreviations (Förkortningar)	Han läste sidor med b la börskurser.
31. Spaces in conjunction with quotation marks (Mellanslag vid citat)	Men Sverige har också " goda möjligheter att "lösa" problemen_"snabbt.
32. Spaces in conjunction with parentheses (Mellanslag vid parenteser)	De nämnde ytterligare några exempel(fascinerande eller hur).
33. Spaces in conjunction with punctuation marks (Mellanslag vid skiljetecken)	Såg du ;hörde du?
34. Spaces in conjunction with special characters (Mellanslag vid specialtecken)	Enligt §2 i bolagsordningen skall stämma sammankallas årligen.
35. Parentheses (Parenteser)	Detta hus {rött (och fult} är gammalt.
36. Number formatting (Sifferformatering)	Summan uppgick till 2,453,995,000.23 dollar.
37. Punctuation marks (Skiljetecken)	Hur kom du hit.
38. Uppercase and lowercase (Stor och liten bokstav)	I alla fall kommer jag i September.
39. Dashes and hyphens (Tankstreck och bindestreck)	Genom det nya samarbetsklimat som Per Olsson eftersträvar --- och som vi förutsätter omfattas av hela regeringen --- bör riksdagsarbetet kunna bli stabilare.
40. Phone numbers (Telefonnummer)	Vi nås på tfn 050 – 524096 efter klockan 19.
41. Colloquialism (Talspråkligt ord)	Enligt filosofien åt direktörn plättar mot vederlag.
42. Archaism (Ålderdomligt ord)	Enligt filosofien åt direktörn plättar mot vederlag.
43. Bureaucratic word (Byråkratiskt ord)	Enligt filosofien åt direktörn plättar mot vederlag.

One could very well discuss whether a smaller number of more general error types could suffice, for instance in the case of syntactic errors, i.e. should one group all the seven or so noun phrase internal error types or the ten verb chain error types each under one single error type. Since each different error type represents a different type of linguistic feature and a different error detection strategy on the part of the grammar checker, it was our assessment that providing more information gives the end-user a better understanding of the inner workings of the grammar checker, which renders the tool less irritating and school-masterly. Furthermore, each error type represents an option that the user can either select in the setup of Grammatifix to be either active or inactive during the grammar checking process. This can be useful either when an end-user deliberately decides to violate certain syntactic, punctuation or number formatting conventions, or when the end-user produces a text type which contains some specific error types that prove to be difficult for the grammar checker to scrutinize with at an acceptable level of precision.

5. Should something happen after error detection?

In principle, it must be said that error detection – with both respectable recall and precision rates – is the theoretically most demanding challenge in creating a grammar checker. In practice, however, error detection, even with a reliable algorithm, must be integrally followed up by support in the treatment of the assumed error to be of real use to a standard end-user (e.g. Domeij et al 1996: 8). First and foremost, the detected error must be diagnosed in such a way that the end-user can understand why a portion of the text has been marked as dubious or erroneous by the grammar checker, so that the end-user may make his or her own educated judgment on the issue, and how this potential error can consequently be corrected. It is the manner in which a grammar checker communicates its linguistic findings to the end-user that the quality of the grammar checker is ultimately perceived by him or her. In the section below, emphasis is put on the treatment of syntactic errors, though the same principles have been applied to the other error types, too.

In the case of Grammatifix, the construction of the error detection algorithms provides a good basis for giving the necessary feedback to the user. As is described by Birn in this volume, the number of Constraint Grammar based error detection rules is manifold to the number of error types, being over 650 rules to the present 26 syntactic error types. Each of these individual error detection rules operate for only one specific error type, and is activated, i.e. the rule flags a detected error, only by a certain sequence of combinations of words and their morphological analyses. Consequently, a rule in fact identifies the specific erroneous word and the erroneous morphosyntactic feature in that word at the same time as the rule detects the entire erroneous construction. As Birn further describes, numerous corpus-based constraints are added to the error detection rules to ensure that an interdependency indeed exists between the words in the assumed construction, so that some particular morphosyntactic characteristic may be validly expected of one of the words that the rule covers, a characteristic which is lacking or wrong when an error is flagged. Thus, the treatment of an error flagging is integrally connected with and determined by the error detection rules. Each error rule can be and is mapped to a specific, formalized error treatment scheme. Several rules, however, may have the same error treatment scheme. Each such error treatment scheme consists of 1) an error heading; 2) a terse error diagnosis text; and 3) an error correction scheme.

The error headings are in fact the same as the names of the error types presented in Tables 1 and 2, and are entry points to the error diagnosis texts. An error diagnosis text conveys the suspect word form, the reason for the suspicion together with the other words involved in the assumed syntactic construction, and finally a description of the necessary correction, if the error detection is indeed judged by the end-user to be correct. The error diagnosis text packs the above information tightly in plain Swedish sentences, which can in free translation be exemplified in the following form: *“Check the word form X. If an A, such as Y, governs a B, then the A should also be in the C form”*. The variable words X and Y above can be directly extracted from the grammar checked text with the help of the error detection rules, whereas the variable words A, B and C are linguistic categories or features, such as *noun* or *genitive*, or combinations of these, determined by the detected error type. Since the underlying linguistic analysis presupposes sentence delimitation, the suspect words can be in addition be presented as marked in their full sentential context.

The error diagnosis texts might be considered relatively heavy and overtly linguistic in wording, but it was found difficult to generate “lighter” phrasings which would have been both accurate enough in describing precisely the intended construction, and relatively short in length. Being concerned with grammar checking, we deemed it natural to use grammatical terms (which are associated with an example word in the error diagnosis text, or explained in a longer explanation text, mentioned below). Nevertheless, the error diagnosis texts are an important part of how end-users perceive Grammatifix and cannot be considered insignificant – quite the contrary. Consequently, research is presently being undertaken at Lingsoft on user reactions to this and other information provided in Grammatifix’ user interface.

Finally, a suggestion for the correction of the suspected erroneous word is provided, when practically feasible. In most error types, the suggestion is generated by applying the error correction scheme, representing the spirit of the error diagnosis text, to the morphological analysis of the erroneous word, in its essence meaning a substitution of the appropriate morphological tags, and then generating the respective new word form. In other error types, the error correction scheme may delete the erroneous word, substitute it with another, generate an erroneously missing word, or reorder an incorrect sequence of words. The basic guideline in the error treatment schemes is to establish a single erroneous component or word, when possible. This works for error types such as gender disagreement where it is self-evident that the head word cannot in practice be erroneous, but for others, such as number disagreement, such a judgement would be fairly difficult without extralinguistic knowledge. In such cases, typically two words are marked as suspect, and consequently two phrases are given as suggestions for correction. Table 3 gives examples of the different types of suggestions for correction that Grammatifix provides. As can be noted, sometimes some non-erroneous words in the context of the erroneous word are included in the suggested change in order to make it easier for the end-user to visualize the suggestion.

Table 3: Examples of the different types of suggestions for the correction of syntactic errors provided by Grammatifix

Suggestion type	**Erroroneous sentence (actual error[s] or error contexts underlined)**	**Suggested change (as it is presented to the end-user)**
One suggested change	I maj i fjol genomgivk Brolin ytterligare <u>ett</u> operation.	ett → en
Two suggested changes	I protest mot <u>de</u> statliga <u>monopolet</u> började han sälja sprit på Drottninggatan i Stockholm.	de statliga monopolet → det statliga monopolet → de statliga monopolen
Deletion	Axelstöd och gymnastik är bästa motmedlen om man inte vill <u>att</u> ha förändringar i nacken och käken som gör spelet stelt.	vill att → vill
Generation of missing word	Vidare ska pengar omfördelas till bland annat satsningar på Internet <u>för stödja</u> myndigheters och företags miljöarbete.	för stödja → för att stödja
Reordering of sequence	Den har setts av så få personer på biograferna att den lär <u>knappast</u> gå över den magiska miljongränsen.	lär knappast → knappast lär

The aforementioned two stages of Grammatifix' implementation, namely error detection and error treatment, with their substages, would seem to correspond to a four-level framework presented by Uszkoreit and adopted in SCARRIE, namely 1) detection; 2) recognition; 3) diagnosis; and 4) correction (where Uszkoreit's 'recognition' rougly corresponds to Grammatifix' error headings) (Uszkoreit 1996, Sågvall Hein 1998). However, Uszkoreit's framework seems to lead to a modular implementation with clear transfer of data from one level to the next, and with different viewpoints or methods at different levels. In contrast, Grammatifix aims at integrating all the stages in *one level*, i.e. an error detection rule specifies deterministically and thus contains implicitly the corresponding error recognition (heading), the error diagnosis and the error correction information. Consequently, after the error detection stage, Grammatifix has all the necessary information to be relayed to the end-user, and no further observation or treatment of the erroneous phrase is necessary.

In addition to the error detection and error treatment modules, a set of longer explanation texts are incorporated into Grammatifix, covering concisely the central syntactic and non-syntactic issues related with the targeted error types. Furthermore, each time after Grammatifix has gone through a selected text, it provides some general statistics, including the number of characters, words, sentences and paragraphs, the average number of characters per words, the average number of words per sentence, and the average number of sentences per paragraph in the checked text. In conjunction with these statistics, Grammatifix also provides a so-called LIX value (*“Läsbarhetsindex”*), which is a readability index developed for Swedish.[9] It should be noted that when used as an integrated module it is up to the client application which components, that have been described in this section and that are provided by Grammatifix, are indeed conveyed to the end-user.

6. Comparison with the other known grammar checkers for Swedish

A comparison of the different error types covered by Grammatifix and the two other publicly known projects, namely SCARRIE and Granska, is presented in Tables 4, 5 and 6 below. The classification is based on Grammatifix in the order as its error types are presented in Tables 1 and 2 above. No qualitative or restrictive judgement has been made on the basis of the example sentences which are provided in the specifications of the other tools. If a description of a particular tool exhibits even a single but clear example of the detection of a particular error type, that error type is marked as covered by the tool. Furthermore, the error types listed in the documentation of the different projects are taken *bona fide*. Consequently, this comparison should be taken with a grain of salt, since the error classifications are not exactly similar, and most certainly have been implemented with varying depth and breadth in the different projects (e.g. Granska: Domeij and Knutsson 1998: 2).

Thus, even though the projects may report exactly the same error type on their feature lists, the individual tools may either detect differing subsets of phrases containing the erroneous feature, and with different levels of syntactic complexity, or may detect these subsets of erroneous phrases in different positions within a sentence. Furthermore, one must note that since neither Grammatifix nor SCARRIE include an integrated spell checking component, purely orthographical errors are lacking from their error types. In the end, this comparison serves probably best as an indication of the varying development foci of the different tools rather than as a definitive evaluation of their coverage or general “goodness”.

As an overview it appears that all the tools aim at covering the basic noun phrase internal and verb chain consistency error types. Granska appears to specialize in a wide range of stylistic evaluation of word use. Compared to Grammatifix, both SCARRIE and especially Granska address the problem of mistakenly writing compound words separately. Grammatifix, on the other hand, seems to have the widest coverage in the punctuation and number formatting errors. Furthermore, even though all the three tools aim in their error treatment schemes at similar goals, i.e. generating replacement suggestions for erroneous words, they differ in their present level of implementation of this function. Presently, Grammatifix and Granska have proceeded the furthest of the three, and have fully implemented error treatment schemes, including correction generation for most error types.

Table 4: A comparison of the syntactic error types of Grammatifix, SCARRIE[10] and Granska[11] (X = evident in example sentences or documentation; x = evident in the Internet demo visited 10.2.2000)

Error type	**Grammatifix**	**SCARRIE**	**Granska**
Definiteness form of noun	X	X	X
Definiteness form of adjective	X	X	X
Number agreement: determiner and noun	X	X	X
Number agreement: adjective and noun	X	X	X
Gender agreement: determiner and noun	X	X	X
Gender agreement: adjective and noun	X	X	X
Masculine form of adjective	X	-	x
Gender agreement: pronoun and noun	X	-	X
Subject complement agreement	X	x	X
Supine without the "ha" auxiliary verb	X	x	X
Double supine	X	X	X
Double passive	X	X	X
S-passive after certain verbs	X	X	x
Infinitive after preposition	X	X	X
Infinitive without an expected "att"	X	x	X
Infinitive with unexpected "att"	X	X	x
Number of finite verbs	X	X	x
No verb	X	X	X
No finite verb	X	X	X
Tense harmony	-	-	X
Position of adverb in subordinate clauses	X	-	X
Position of negated element in subordinate clauses	X	x	x
Constituent order in subordinate interrogative clauses	X	-	X
Constituent order in the beginning of an inverted main clause	-	X	-
Double negation	X	-	x
Use of preposition with two-part conjunctions	X	-	x
Form of pronoun after preposition	X	x	X
The construction "möjligast" + adjective	X	x	x
Repeated words	(spell checker)	X	x
Compound words mistakenly written separately	-	X	X
Words written mistakenly together	(spell checker)	-	X
Incorrect preposition in conjunction with a fixed expression	-	-	X

Table 5: A comparison of the punctuation and number formatting violation error types of Grammatifix, SCARRIE[10] and Granska[11] (X = evident in example sentences or documentation; x = evident in the Internet demo visited 10.2.2000)

Error type	**Grammatifix**	**SCARRIE**	**Granska**
Quotation marks	X	-	-
Date expressions	X	x	X
Several spaces in a row	X	x	-
Multipart abbreviations	X	x	X
Spaces in conjunction with quotation marks	X	-	-
Spaces in conjunction with parentheses	X	x	-
Spaces in conjunction with punctuation marks	X	x	X
Spaces in conjunction with special characters	X	x	x
Parentheses	X	-	-
Number formatting	X	x	X
Punctuation marks	X	-	X
Uppercase and lowercase	X	x	-
Dashes and hyphens	X	-	X
Phone numbers	X	x	-
Mixing of uppercase and lowercase within words	(spell checker)	(spell-checker)	X

Table 6: A comparison of the stylistic evaluation error types of Grammatifix, SCARRIE[10] and Granska[11] (X = evident in example sentences or documentation; x = evident in the Internet demo visited 10.2.2000)

Error type	**Grammatifix**	**SCARRIE**	**Granska**
Colloquialisms	X	-	X
Archaisms	X	-	X
Bureaucratic word	X	-	X
Abstract words	? (=bureaucratic)	-	X (Long verb forms, compound verbs and long verb forms are included here under archaisms)
Non-recommended computer terms	-	-	X
Tautological expressions	-	-	X
Contaminated constructions	-	-	X
Foreign words with difficult inflection	x	x	X
Conjunctions as first words in sentences	-	-	X
Use of impersonal subject	-	-	? (=inactivated)
Unnecessary nominalization	-	-	? (=inactivated)
Difficult words according to *Språkklyftan*	-	-	? (=inactivated)
Words which have a different meaning in standard vs. bureaucratic contexts	-	-	? (=inactivated)

This overview is not, of course, a check list that can as such be used to select one solution over another or argue for or against one solution at the present time. It should be remembered that many of the error types listed in the overview have only been partially implemented in the various solutions. Furthermore, there are numerous possible error types that are not listed in the overview, which might nevertheless be of considerable value to end-users, even more than some of the error types presently on the list. Taking into consideration the sheer magnitude of possibilities in grammar checking as laid forth in section 2, and personal experience of how much effort has gone into getting only a single solution, Grammatifix, at the level it now is, it is indeed a very interesting question in what directions these three, and possibly some other, new solutions will develop in the coming years.

Notes

[1] I am indebted to the entire product development team who undertook the man-months of practical work to make this project happen: Jussi Birn, Mathias Creutz, Era Eriksson, Risto Kankkunen, Ari Paavilainen, Alexander Paile Pasi Ryhänen, and Fredrik Westerlund. Furthermore, I am thankful for Jussi Birn for proof-reading this paper on several occasions and providing insightful comments.

[2] Personal communications from Petteri Suoranta and Kaarina Hyvönen, successive product development managers at Kielikone Ltd.

[3] Personal communication from Peter Bursell, formerly of Norstedts Publishers, who participated in the project

[4] The term syntactic (syntax) is chosen here instead of grammar, since syntax specifically refers to relationships and constructions between words whereas grammar (grammatical) is often used to cover the general structure of a language, including morphology.

[5] Granska has since abandoned this principle and is presently designed as an interactive tool.

[6] As set forth in the section ‘Basic Functions’ of the Project Summary of the SCARRIE project: http://www.scarrie.com/

[7] Two of the project members had Swedish as their mother tongue.

[8] Examples of error types that were deemed difficult to detect in general were error types that would seem to require a full sentential analysis or even more in order to be reliable, such as ambiguity or unclear reference of pronouns or prepositions within or between sentences, errors in ellipses, and analysis of sentential integrity.

[9] The LIX value, attributed to be developed by Erik Björnsson in the 1960’s, is calculated with the formula $LIX = average(N(i)) + 100 \times average(L(i)/N(i))$, where $N(i)$ the number of words in sentence i and $L(i)$ is the number of words with more than seven characters in sentence i, calculated for all the sentences i in the text.

[10] The sources of the error type listing for SCARRIE are <http://stp.ling.uu.se/~ljo/scarrie-pub/scarrie_examples_sv.html> and the Internet demo <http://stp.ling.uu.se/~ljo/scarrie-pub/scarrie_sv.html>, visited on 10.2.2000.

[11] The sources of the error type listing for Granska are <http://www.nada.kth.se/theory/projects/granska/rapporter/grammatikregler.html>, the Internet demo <http://www.nada.kth.se/theory/projects/granska/demo.html>, visited 10.2.2000 and Domeij & Knutsson 1998.

References:

Arppe, A; Birn, J; Westerlund, F. 1999. *Lingsoft’s Swedish Grammar Checker*. http://www.lingsoft.fi/doc/swegc/

Bernth, A. 1997. EasyEnglish: A Tool for Improving Document Quality. *The Proceedings of the Fifth Conference on Applied Language Processing*, Washington, 159-165.

Birn, J. (this volume). Detecting grammar errors with Lingsoft's Swedish grammar checker.

Birn, J. 1998. Swedish Constraint Grammar: A short presentation. http://www.lingsoft.fi/doc/swecg/

Bustamante, F. R. & and Léon, F. S. 1996. GramCheck: A Grammar and Style Checker. *The Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, 175-181.

Domeij, R; Knutsson, O; Larsson, S. 1996. *Datorstöd för språklig granskning under skrivprocessen: en lägesrapport. Report number IPLab-109*. Stockholm: Interaction and Presentation Laboratory, Department of Numerical Analysis and Computer Science, Royal Institute of Technology.

Domeij, R; Knutsson, O; Larsson, S; Severinsson Eklundh, K; Rex, Å. 1998. *Granskaprojektet 1996-1997. Report number IPLab-146*. Stockholm: Interaction and Presentation Laboratory, Department of Numerical Analysis and Computer Science, Royal Institute of Technology.

Domeij, R; Knutsson, O. 1998. *Granskaprojektet: Rapport från arbetet med granskningsregler och kommentarer. Internal subreport 26.8.1998*. Stockholm: Interaction and Presentation Laboratory, Department of Numerical Analysis and Computer Science, Royal Institute of Technology.

Karlsson, F. 1992. SWETWOL: Comprehensive Morphological Analyzer for Swedish. *Nordic Journal of Linguistics,* volume 15, 1992, 1-45.

Severinsson Eklundh, K. 1993. Skrivprocessen och datorn. Contribution to *Människor-Datateknik-Arbetsliv* (Ed. Lennart Lennerlöf).

Sågvall Hein, A. 1998. A Chart-Based Framework for Grammar Checking: Initial Studies. *Proceedings of 11th Nordic Conference on Computational Linguistics,* Copenhagen, 68-80.

Uszkoreit, H. 1996. Grammar Checking. Theory, Practice and Lessons learned in LATESLAV. Concluding oral presentation at the final review meeting of the *Lateslav Project (PECO 2824),* Prague, August 1996. As presented in Sågvall Hein, Anna 1998.