Granska API – an Online API for Grammar Checking and Other NLP Services

Jonas Sjöbergh
Theoretical Computer Science, KTH
Stockholm, Sweden
jsh@kth.se

Viggo Kann
Theoretical Computer Science, KTH
Stockholm, Sweden
viggo@nada.kth.se

Abstract

We present an online API to access a number of Natural Language Processing services developed at KTH[1]. The services work on Swedish text. They include tokenization, part-of-speech tagging, shallow parsing, compound word analysis, word inflection, lemmatization, spelling error detection and correction, grammar checking, and more. The services can be accessed in several ways, including a RESTful interface, direct socket communication, and premade Web forms. The services are open to anyone. The source code is also freely available making it possible to set up another server or run the tools locally. We have also evaluated the performance of several of the services and compared them to other available systems. Both the precision and the recall for the Granska grammar checker are higher than for both *Microsoft Word* and *Google Docs*. The evaluation also shows that the recall is greatly improved when combining all the grammar checking services in the API, compared to any one method, and combining services is made easy by the API.

1 Introduction

A number of Natural Language Processing (NLP) tools for analysis of Swedish text have been developed at KTH (Kann, 2010). Most of the tools were developed in projects focused on grammar checking: "Algoritmer för svenska språkverktyg" ("Algorithms for Swedish language tools"), "Svensk grammatikgranskning" ("Swedish grammar checking"), and "CrossCheck – svensk grammatikkontroll för andraspråksskribenter" ("CrossCheck – Swedish grammar checking for second language writers"). This lead to a focus on tools that are useful for grammar checking, but low level language analysis tools useful in other applications are also included.

[1]A short version of this paper was presented at the SLTC-2020 conference.

The source code for the tools has been freely available since the tools were developed, and anyone is free to install and use them locally. Now we have also made an online API available[2]. It can be used to access the tools running as services on a server at KTH, and these services are also open for anyone to use. They can be used by a user by hand, typing text or copying text from some other program, and by programs using the services to do some analysis they need.

We have also built an example application that uses the services to create a graphical text exploration environment, and we have evaluated some of the tools provided in the API, comparing them to other available systems that perform the same service.

2 Available Services

The available services can be divided into three types of services: low-level or preprocessing NLP tools that can be used to build more advanced services, tools to help when developing and evaluating NLP tools, and high-level NLP services that are directly useful to end users. It is still possible to build new tools on top of the high-level services.

The low-level services in the Granska API are:

Tokenization The Granska tokenizer tokenizes text into words and sentences. It is integrated in the Granska tagger and in the Granska grammar checker below, but it is also possible to build a stand-alone tokenizer.

PoS Tagging The Granska tagger (Carlberger and Kann, 1999) does part-of-speech tagging of Swedish text. It is a Hidden Markov Model tagger trained on the SUC corpus (Ejerhed et al., 1992) using a slightly modified version of the SUC tag set. The tagger is integrated

[2]`https://skrutten.csc.kth.se/granskaapi/`

in the Granska grammar checker but can also run as a stand-alone application.

PoS Tagging without context Taggstava (Kann, 2010) is a tagger that assigns part-of-speech tags to words without using context information. It uses inflection rules for Swedish to determine what inflected form a word could be. No disambiguation is done for ambiguous words, all possible tags are returned. Taggstava uses the same rules and reference data as the spelling error detection program Stava below.

Shallow Parsing The GTA parser (Knutsson et al., 2003) does shallow parsing of Swedish text based on hand written rules. It identifies clause boundaries and phrases. The internal structures of phrases are identified, e.g. a noun phrase being part of a prepositional phrase, but a full tree for the whole sentence is not built.

GTA is built to be robust to noisy data (i.e. text with many errors) since it is built for and used in the grammar checker Granska below, which is expected to run on texts with possibly very many errors in them.

For convenience, there are also two services that return subsets of the GTA information, one that returns only clause boundaries, and one that returns only the phrase structure.

Compound Word Analysis SärStava (Sjöbergh and Kann, 2004) is a tool that gives the most likely interpretation of a compound word, or all possible interpretations. Possible interpretations are found using the Stava compound word analysis methods. Then statistical data and some heuristics are used to decide which interpretation is most likely for ambiguous compounds. No methods using the context of the word are used, though.

Word Inflection The Granska Inflector inflects Swedish words. It can generate a specific inflected form or a list of all possible inflections.

Lemmatization This service uses the Granska tagger to find the lemma form of words.

Word-Tag-Lemma Several other services expect the input to be triples of word, part-of-speech tag, and lemma form of the word. For convenience, a service that takes plain text and provides word-tag-lemma triples, by calling the Granska tagger, is also provided.

There is currently only one service in the development and evaluation tools category. Other tools for evaluating NLP tools are available to run locally, but have not been made available as online services yet. The available tool is:

Realistic Spelling Error Generation Missplel (Bigert et al., 2003) is a tool that automatically inserts spelling errors in texts. Different types of errors can be simulated, for example keyboard mistypes where a neighboring key is pressed by mistake or sound-alike errors where the writer may not know the correct spelling of a word they know how to say.

Missplel can be used to automatically evaluate the robustness of other NLP systems by showing how the performance degrades when there are errors in the text. For example, an evaluation can be done by running a parser on a test text and then running it on the same text with added errors. Ideally, the parser should produce similar output the second time, since the "intended" meaning of the text is the same. This way the robustness can be evaluated without using any annotated data.

The high-level services are all spelling and grammar checking services, since the tools were built in research projects focused on this. The available services are:

Spelling Error Detection and Correction Stava (Domeij et al., 1994) is a very powerful spelling correction tool for Swedish that finds spelling errors and suggests corrections. Stava handles the very productive compounding in Swedish using rules for how compounds can and cannot be created in Swedish. The compound analysis can also be accessed separately, as mentioned above. In Swedish it is very common to create new compound words in normal text, and without some form of compounding analysis there are normally very many false alarms from spelling error detection tools.

Grammar Checking using Rules The Granska (Domeij et al., 2000) system detects grammatical errors in Swedish text based on manually

Goal: get the most likely interpretation of the compound “glasstrut”.
API call: `https://skrutten.csc.kth.se/granskaapi/compound/best/glasstrut`
Output: `glasstrut glass|strut`

Goal: get all possible interpretations of the compound “glasstrut”, in JSON.
API call: `https://skrutten.csc.kth.se/granskaapi/compound/json/all/glasstrut`
Output: `["word":"glasstrut", "parts":["glas|strut", "glass|strut", "glass|trut"]]`

Goal: get phrase structure in the sentence “GTA kan analysera svensk text.”.
API call: `http://skrutten.csc.kth.se/granskaapi/chunk?text=GTA+kan+analysera+svensk+text+.`
Output: `GTA NPB, kan VCB, analysera VCI, svensk APMINB|NPB, text NPI, .  0`

Figure 1: Example API calls and the corresponding outputs

written error detection rules. The rule language (Knutsson et al., 2001) is quite powerful and the rule writer has access to all the information provided by the tools mentioned above. Rules can for example be written to allow suspicious things if they cross a phrase boundary (to reduce false alarms), or to change the inflected form of a suspicious word to a form more suitable to the surrounding context using the inflector above, etc.

Extra rules can be added to each API call. These can be used to detect new types of errors not covered by the standard rules or to influence the behavior of Granska (e.g. by adding more parsing rules). Here is an example of a simple rule:

```
altcorr@kong{
 X(wordcl=dt),
 Y(wordcl=nn & num!=X.num)
  -->
 corr(X.form(num:=Y.num))
 corr(Y.form(num:=X.num))
 action(scrutinizing)}
```

This rule finds places where a determiner (word class is “dt”) is followed by a noun (word class “nn”), but they have different number, i.e. it finds agreement errors since determiners and nouns should normally have the same number in Swedish. It then suggests two possible corrections, changing the number of the determiner or changing the number of the noun. The suggested corrections are generated with the inflector above.

Grammar Checking using PoS n-grams
ProbCheck (Bigert and Knutsson, 2002) detects grammatical errors in text using statistical analysis of part-of-speech n-grams. Based on n-gram statistics from correct text, it finds part-of-speech sequences that are rare in the reference data. It also uses the GTA parser above, since phrase and clause boundaries can cause very rare PoS n-grams even in correct text and thus lead to false alarms. ProbCheck usually runs integrated in Granska but running only ProbCheck is also possible.

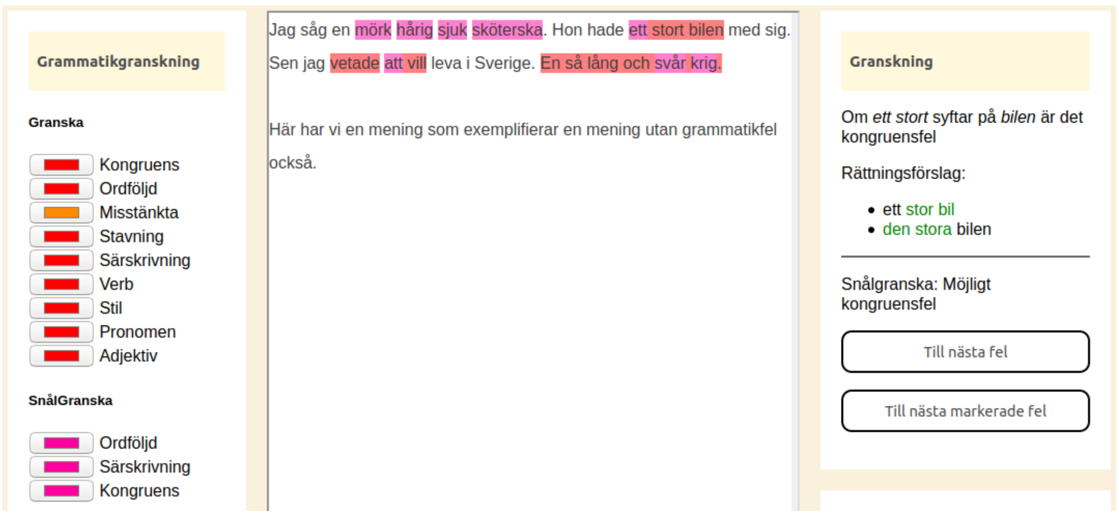
ProbCheck was created in a project focused on helping second language learners. Learners of a language make many unpredictable errors that it can be hard to write error detection rules for. There are also generally a lot of errors, and thus not much correct text as context to base error detection rules on.

Grammar Checking using Machine Learning
SnålGranska (Sjöbergh and Knutsson, 2005) detects grammatical errors using machine learning trained on texts with synthetic errors added. By itself it does not perform as well as Granska, but it does detect errors that Granska does not detect, and it is possible to use both systems together to get improved coverage (Bigert et al., 2004).

3 Ways to Access the Services

All the services mentioned in the previous section can be accessed online. There are simple Web forms where you can enter words or text by hand (or by copy-paste from other applications) and see what the tools can do.

There is also a RESTful API to access the services. This allows typing in requests in the URL bar of a Web browser by hand, but is mainly intended for other programs to automatically use the services for language processing tasks they may need. Most


Grammatikgranskning

Granska

- 🟥 Kongruens
- 🟥 Ordföljd
- 🟧 Misstänkta
- 🟥 Stavning
- 🟥 Särskrivning
- 🟥 Verb
- 🟥 Stil
- 🟥 Pronomen
- 🟥 Adjektiv

SnålGranska

- 🟪 Ordföljd
- 🟪 Särskrivning
- 🟪 Kongruens

Jag såg en mörk hårig sjuk sköterska. Hon hade ett stort bilen med sig. Sen jag vetade att vill leva i Sverige. En så lång och svår krig.

Här har vi en mening som exemplifierar en mening utan grammatikfel också.

Granskning

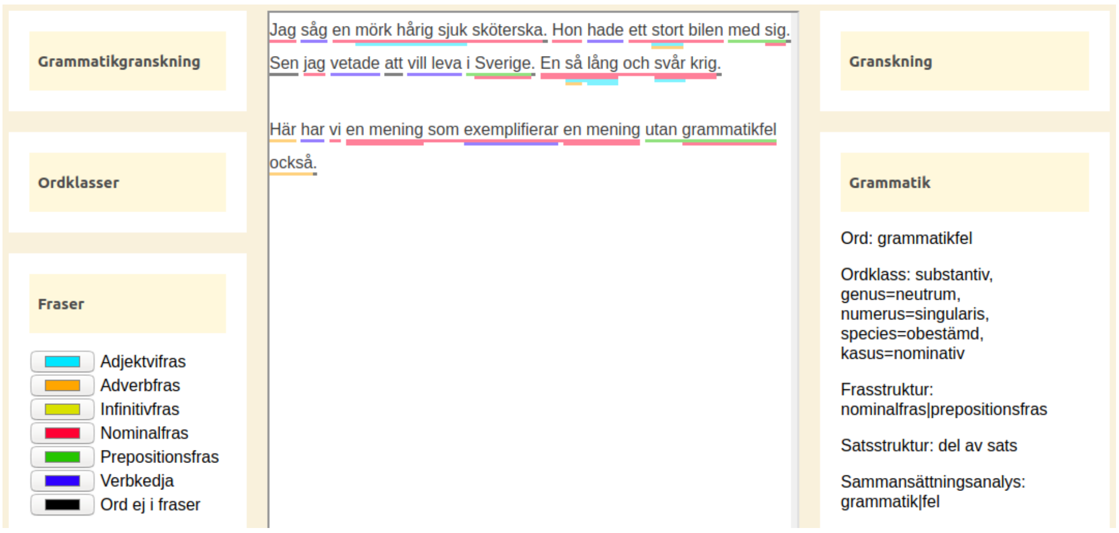
Om *ett stort* syftar på *bilen* är det kongruensfel

Rättningsförslag:

- ett stor bil
- den stora bilen

Snålgranska: Möjligt kongruensfel

Figure 2: FörHandsGranska, built on top of the API services. Here working as a text editor with spelling and grammar checking support, letting the user use suggested corrections through simple clicks.


Grammatikgranskning

Ordklasser

Fraser

- 🟦 Adjektivfras
- 🟧 Adverbfras
- 🟨 Infinitivfras
- 🟥 Nominalfras
- 🟩 Prepositionsfras
- 🟦 Verbkedja
- ⬛ Ord ej i fraser

Jag såg en mörk hårig sjuk sköterska. Hon hade ett stort bilen med sig. Sen jag vetade att vill leva i Sverige. En så lång och svår krig.

Här har vi en mening som exemplifierar en mening utan grammatikfel också.

Granskning

Grammatik

Ord: grammatikfel

Ordklass: substantiv, genus=neutrum, numerus=singularis, species=obestämd, kasus=nominativ

Frasstruktur: nominalfras|prepositionsfras

Satsstruktur: del av sats

Sammansättningsanalys: grammatik|fel

Figure 3: FörHandsGranska, showing linguistic analysis. Words can be colored based on part-of-speech, phrase structure, or clause structure. Compound word analysis, possible inflections, etc., are also shown.

services can send back the reply in either plain text form, HTML, JSON, or XML. Figure 1 shows example API calls and the corresponding outputs.

It is also possible to access the services using socket communication. When communicating with the services directly using a socket, most services will only return the raw output of the original tool (for example not provide the result in JSON).

If no input is given, each service will display a Web page with information on how to call the service. An example Web form that uses the service is shown, and this can be used as a reference to see what input is expected, how to format the input, etc. A few example words or sentences are also provided to give a quick overview of what typical input and output can be expected to look like.

The API allows building new tools based on the services, creating new interfaces to the services, or integrating the services into existing tools (e.g. an editor or word processor). If an online service is not suitable, for example for a system that is expected to run offline, the source code for all the tools is also freely available. This makes it possible to install any tool and run it locally, or to install tools and set up a new server that can provide the same services.

4 Example Application based on the Services

We have created an example application using a number of the services described above.

System	True Pos.	Diag. Errors	False Pos.	Total Reports	False Neg.	Precision (%)	Pseudo Recall (%)
Granska	211	4	203	414	297	51	42
ProbCheck	130	-	468	598	378	22	26
SnålGranska	112	103	736	848	396	13	22
All Granska API	421	86	1262	1683	87	25	83
MS Word	64	2	270	334	444	19	13
Google Docs	107	5	663	770	401	14	21

Table 1: Evaluation on text with few errors (published novels). 508 errors annotated in 101,279 tokens. Pseudo recall (and False Negatives) is calculated based on all the errors found by any system, but since there are also errors not found by any system the true recall is lower. "Diag. Errors" are error reports where there is an error in the text but the diagnosis is wrong, for example reporting an agreement error when it is actually a spelling error. ProbCheck does not give error diagnoses.

System	True Pos.	Diag. Errors	False Pos.	Total Reports	False Neg.	Precision (%)	Pseudo Recall (%)
Granska	978	85	581	1559	888	63	52
ProbCheck	341	-	507	848	1525	40	18
SnålGranska	497	428	763	1260	1369	39	27
All Granska API	1579	374	1694	3273	287	48	85
MS Word	562	50	376	938	1304	60	30
Google Docs	360	8	407	767	1506	47	19

Table 2: Evaluation on blog texts, 1,866 errors annotated in 97,645 tokens. Pseudo recall (and False Negatives) is calculated based on all the errors found by any system, but since there are also errors not found by any system the true recall is lower. "Diag. Errors" are error reports where there is an error in the text but the diagnosis is wrong, for example reporting an agreement error when it is actually a spelling error. ProbCheck does not give error diagnoses.

FörHandsGranska[3] is a graphical text exploration tool. It can mark writing errors in different colors and suggest corrections, working as an editor with built in spelling and grammar checking tools. Errors in the text can be replaced with corrected text by simply clicking on suggestions from the grammar checking tools.

It can also add linguistic markup, coloring words based on their part-of-speech, underlining different types of phrases in different colors, or show clause boundaries. It also shows all inflections of a word, the compound analysis of compound words, and more. In this way, it can be used as a linguistic exploration tool or language learning tool. Interaction is also possible through for example clicking on a listed inflected form to change the inflection of the word in the original text.

It is possible to show both suspected writing errors and linguistic markup at the same time. Adding more rules in the Granska rule language is also supported. Two example screenshots of FörHandsGranska are shown in Figures 2 and 3.

FörHandsGranska also allows quick lookup in other online services not provided by the Granska API, such as the *SAOB* dictionary, the *Lexin* search service, or concordance lookup in the *Korp* service.

FörHandsGranska is written in JavaScript and is basically a graphical interface that calls the services of the Granska API when language analysis is needed.

5 Evaluation

There have been many evaluations of the different tools. For evaluations of the individual tools, we refer to the respective publications cited above.

We have also done a new evaluation using the tools through the Granska API. We have evaluated the Granska grammar checking tool on Swedish text. Since Granska also uses almost all of the other tools in the API, this gives an overview of how well all the tools can work together.

We fed unannotated Swedish text to the Granska API. For comparison, we also fed the same text

[3]https://skrutten.csc.kth.se/fhg/

to the spelling and grammar checking tool integrated in *Microsoft Word 2016*, the spelling and grammar checking tool in *Google Docs*, and the grammar checkers ProbCheck and SnålGranska. We also combined all the grammar checking services in the Granska API (Granska, ProbCheck, and SnålGranska) as one grammar checking service to see how much the recall improves by using several methods that hopefully complement each other.

All error reports from the grammar checkers were manually annotated as correct or not, but we did not manually check the text for errors not found by any of the grammar checking tools. A quick manual check of a small sample of the evaluation data showed that there are indeed errors that are missed by all the grammar checking tools.

The evaluation texts used all come from the *Språkbanken* corpus resources[4]. There are many corpora available for download, and there is a search interface with NLP tools that can be used to search the available corpora (Borin et al., 2012).

Table 1 shows the evaluation results on texts with few errors to find, in this case texts from published novels. Since there are few true errors to be found, precision can be expected to be low.

Table 2 shows the evaluation results on blog texts, which have more errors than the published novels. As expected, all grammar checking methods achieve higher precision in this test set.

The results support the idea that the different grammar checkers complement each other (as mentioned in Section 2, ProbCheck was explicitly created to complement Granska) since no grammar checker found even half of the total errors in the published novels and only one system found just over half the errors in the blog texts, when compared to all errors found by all the systems in total.

Using the Granska API it is easy to combine the output from any system included in the API. Combining the three services provided in the Granska API gives much higher recall than any single system achieves, as seen in Tables 1 and 2, though the precision is of course lower than the precision of the highest performing individual system.

The results also indicate that the grammar checking methods in the Granska API perform competitively when compared to other grammar checking systems for Swedish. Both the precision and the recall of the Granska grammar checker is higher than those of the grammar checking methods in both *Microsoft Word* and *Google Docs* in these test sets.

[4]https://spraakbanken.gu.se/en/resources

6 Related Work

There are other NLP APIs, both online APIs and APIs for using tools locally. Most APIs are for English but APIs for other languages are also available.

For Swedish, the *Sparv* corpus annotation pipeline (Borin et al., 2016) has an online API. It supports tokenization, lemmatization, part-of-speech tagging, compound analysis, dependency parsing, named entity recognition, and more. *Sparv* also supports languages other than Swedish.

The *SVENSK* project (Gambäck and Olsson, 2000) collected NLP tools for Swedish, including part-of-speech tagging, parsing, text classification, and more. Resources from different sources were integrated into one consistent framework using *GATE* (Cunningham et al., 1996).

7 Conclusions

We provide an online API to access NLP services for Swedish text. Both low level services like part-of-speech tagging and high level services like grammar checking are provided. The services are freely available online, with several ways to access them. The source code is also freely available, allowing users to set up their own servers or run the tools locally. The tools can be used by hand or integrated in other programs. As an example of what can be done by using the API, we have also created an online application for interactive text exploration that uses the API for all linguistic analysis needed.

We evaluated some of the higher level services, that in turn use most of the low level services, comparing them to other available systems. The results show that the performance is improved by combining several services, and that the provided services in themselves perform competitively compared to other available systems. Combining systems is easy using the provided API.

References

Johnny Bigert, Linus Ericson, and Antoine Solis. 2003. Missplel and AutoEval: Two generic tools for automatic evaluation. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland.

Johnny Bigert, Viggo Kann, Ola Knutsson, and Jonas Sjöbergh. 2004. Grammar checking for Swedish second language learners. In Peter Juel Henrichsen, editor, *CALL for the Nordic Languages*, pages 33–47. Samfundslitteratur.

Johnny Bigert and Ola Knutsson. 2002. Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings of Romand 2002, Robust Methods in Analysis of Natural Language Data*, pages 10–19, Frascati, Italy.

Lars Borin, Markus Forsberg, Martin Hammarstedt, Dan Rosén, Roland Schäfer, and Anne Schumacher. 2016. Sparv: Språkbanken's corpus annotation pipeline infrastructure. In *Proceedings of SLTC 2016*, Umeå, Sweden.

Lars Borin, Markus Forsberg, and Johan Roxendal. 2012. Korp - the corpus infrastructure of Språkbanken. In *Proceedings of LREC 2012*, pages 474–478, Istanbul, Turkey.

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

Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. 1996. GATE – a general architecture for text engineering. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Richard Domeij, Ola Knutsson, Johan Carlberger, and Viggo Kann. 2000. Granska – an efficient hybrid system for Swedish grammar checking. In *Proceedings of Nodalida '99*, pages 49–56, Trondheim, Norway.

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

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå Corpus project. Technical report, Department of General Linguistics, University of Umeå (DGL-UUM-R-33), Umeå, Sweden.

Björn Gambäck and Fredrik Olsson. 2000. Experiences of language engineering algorithm reuse. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece.

Viggo Kann. 2010. KTHs morfologiska och lexikografiska verktyg och resurser (Morphological and lexicographical tools and resources from KTH). *LexicoNordica*, 17:99–117. QC 20120126.

Ola Knutsson, Johnny Bigert, and Viggo Kann. 2003. A robust shallow parser for Swedish. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland.

Ola Knutsson, Johan Carlberger, and Viggo Kann. 2001. An object-oriented rule language for high-level text processing. In *NoDaLiDa'01 - 13th Nordic Conference on Computational Linguistics*, Uppsala, Sweden.

Jonas Sjöbergh and Viggo Kann. 2004. Finding the correct interpretation of Swedish compounds a statistical approach. In *Proceedings of LREC-2004*, pages 899–902, Lisbon, Portugal.

Jonas Sjöbergh and Ola Knutsson. 2005. Faking errors to avoid making errors: Very weakly supervised learning for error detection in writing. In *Proceedings of RANLP 2005*, pages 506–512, Borovets, Bulgaria.