NEALT PROCEEDINGS SERIES
VOL. 3

Proceedings of the Workshop on NLP for Reading and Writing – Resources, Algorithms and Tools

November 20, 2008

Stockholm, Sweden

SLTC 2008

Editors

Rickard Domeij
Sofie Johansson Kokkinakis
Ola Knutsson
Sylvana Sofkova Hashemi

NORTHERN EUROPEAN ASSOCIATION FOR LANGUAGE TECHNOLOGY

Proceedings of the Workshop on NLP for Reading and Writing
– Resources, Algorithms and Tools
NEALT Proceedings Series, Vol. 3

ISSN 1736-6305

Published by
Northern European Association for Language
Technology (NEALT)
http://omilia.uio.no/nealt

Electronically published at
Tartu University Library (Estonia)
http://dspace.utlib.ee/dspace/handle/ 10062/4116

Volume Editors
Rickard Domeij
Sofie Johansson Kokkinakis
Ola Knutsson
Sylvana Sofkova Hashemi

Contents

Preface

New tools and media are creating new possibilities for people to communicate and find information over the Internet in a multilingual, global society. This is reflected in the fact that writers now write more in everyday life, typically create text directly on the computer and on the Internet (e.g. on blogs and other Web 2.0 technologies), and use new forms of writing in e-mails, web chats or SMS-messages on mobile phones. This concerns not only adults at work, but also children and youth.

However, the new possibilities also put higher demands on our skills in using language for achieving our communicative goals in different situations. If the demands are not met, a great number of people will risk being excluded from participation in the emerging information society. Of course, this will have unfortunate consequences not only for the individual, but also for society at large. Therefore, there is a growing need for support in reading and writing.

Certainly, there already exists writing tools such as authoring aids in word processors and instructional support in writing education. However, existing solutions have great difficulties meeting the growing need of support. In part, this is due to technical limitations of the tools in processing language, but also to the fact that these tools are poorly adjusted to different tasks, target user groups, genres and media.

Another unfortunate fact is that both research activities and commercial improvements have been lacking until recently. During the last years, the situation has changed to some extent, and several workshops at the largest NLP-conferences have included this field. A short review of workshop proceedings shows that the methods used are based on machine-learning, while rule-based methods are more or less abandoned. This raises a question: Is this due to limited economical resources, or are grammar checking methods based on machine-learning the future for the field? Apart from spelling and grammar checking, many other technologies are knocking on the door, including for instance information retrieval, and speech technology.

In short, the communicative situation and the role of written language have changed, and so has the development of NLP-techniques; this brings new technological possibilities to this field. The purpose of the workshop was to gather people with experience in developing, studying and using such NLP-tools, and to discuss how they can be developed further in order to better meet the needs of language users today and tomorrow – at work, in school or elsewhere.

Eight papers were submitted for review. Four of them were accepted. They were approved in consensus after having been reviewed and ranked by each of the reviewers: Rickard Domeij, Sofie Johansson Kokkinakis, Ola Knutsson and Sylvana Sofkova Hashemi. After revision, the papers are now published in these proceedings together with a written version of Koenraad de Smedt's invited speech.

We were both surprised and pleased about the interest for the workshop from contributors and visitors. It confirms the impression that NLP for reading and writing now gets more attention again after a period of little interest. As exemplified by the contents of these proceedings, there is a wide spectrum of interesting and promising work going on for the benefit of the readers and writers of tomorrow.

Rickard Domeij, Sofie Johansson Kokkinakis,
Ola Karlsson, Ola Knutsson and Sylvana Sofkova Hashemi

Workshop on NLP for Reading and Writing – Resources, Algorithms and Tools

November 20, 2008, Stockholm, Sweden

Workshop webpage:
http://spraakbanken.gu.se/personal/sofie/SLTC_2008/SLTC_2008.html

Conference webpage:
http://www.speech.kth.se/sltc2008/

INVITED SPEAKER

Koenraad de Smedt, Univ. of Bergen, Norway.

ORGANIZERS

Rickard Domeij (rickard.domeij@sprakradet.se)
Language Council of Sweden, Institute of Language and Folklore

Sofie Johansson Kokkinakis (sofie@svenska.gu.se)
Institute for Swedish as a Second Language, Språkbanken, Department of Swedish, University of Gothenburg

Ola Karlsson (ola.karlsson@sprakradet.se)
Language Council of Sweden, Institute of Language and Folklore

Ola Knutsson (knutsson@csc.kth.se)
School of Computer Science and Communication, KTH

Sylvana Sofkova Hashemi (sylvana@ling.gu.se)
Department of Linguistics, University of Gothenburg

NLP for writing: What has changed?

Koenraad De Smedt (University of Bergen)

Workshop on NLP for reading and writing — Resources, algorithms and tools. Stockholm, Nov. 20, 2008

It might appear that few advances have been made in proofreading technology since the 1980s[1]. On the one hand, spelling and grammar checking have become standard features in many kinds of applications that involve writing. On the other hand, a number of advanced research ideas and results from the 1980s do not seem to have been applied or further pursued in newer research. While there is continued research activity in the area of NLP for writing, the scale of projects in this area is not what it used to be. The present moment is therefore an opportunity to look back and reflect on what has been done so far and what has changed[2].

In the 1980s, several academic and commercial research groups in NLP started to turn their attention to automatic proofreading or *text critiquing*. One of the earliest large scale projects was the Writer's Workbench (Macdonald et al., 1982), followed by IBM's EPISTLE project (Heidorn et al., 1982), continued as CRITIQUE (Richardson and Braden-Harder, 1988), which was intended to check and correct the spelling, grammar and style of business letters in English. CRITIQUE uses a parser and grammar of English with relaxation and backoff, and applied lexical substitution to easily confused words. Figures 1 and 2 present screenshots from IBM terminals showing CRITIQUE feedback on mistakes in a business letter.

ESPRIT project OS-82 'Intelligent Workstation' was one of the earliest European applied IT projects that included the development of a proofreading tool. Under the name Author Environment, the tool was targeted at business letters in Dutch and English. Like CRITIQUE, Intelligent Workstation used a grammar and a parser with relaxation to correct grammatical

[1]In his summary submitted to the present workshop, Sjur Nørstebø Moshagen writes *"Utviklinga av grunnleggjande språkteknologiske verkty for vanlege brukarar, slik som gode stavekontrollar og presis orddeling, har i praksis ikkje gått framover sidan 1980-talet."*

[2]The present contribution has a limited scope and does not intend to present an encompassing overview of past work.

errors. It combined grapheme-to-phoneme conversion with trigrams so as to find similar-sounding spellings (van Berkel and De Smedt, 1988) and it provided single-click consultation of a dictionary and encyclopedia. The most advanced functionality consisted of the production of textual variants, not only by finding synonyms and related words, but also by changing from singular to plural and from active to passive and vice versa. The necessary changes were propagated throughout the document by means of a *grammar spreadsheet*. Figures 3 to 6 show examples of interaction with the Author Environment.

In the 1990s, some new techniques were explored and new insights were gained. Vosse (1994) built further on some techniques from OS-82, resulting in the comprehensive CORRIE system for Dutch spelling and grammar checking, which was also used as the basis for the SCARRIE project, supported by the European Commission and aimed at Danish, Norwegian and Swedish. Both CORRIE and SCARRIE offer advanced compound analysis, which is very important for the targeted languages. Parsing at sentence level was also included and functional, but the parser was not disambiguating, so that the number of ambiguities in authentic text remain a problem. GRANSKA (Domeij et al., 2000) for Swedish concentrated on grammar checking, using an HMM disambiguating tagger, tokenizer and rules, and generated a lot of exciting research, not only on techniques but also on user acceptance.

In the 1990s, commercialization by Microsoft, Lingsoft and other companies began to take a hold. Microsoft developed a grammar API and started to provide comments through red squiggles, dialogue boxes and the now discontinued paperclip 'Clippy' with a speech bubble. However, part of the targeted application area was moving faster than the technology. By the turn of the millennium, the typing of business letters was no longer a major office chore. Today, formal business letters have to some extent been replaced by communication through new channels such as email and web-based interaction, while also SMS must be mentioned as a new medium and voice input is starting to become a plausible option. The need for basic spelling and grammar checking remains, so that these functions have also become available in email and browser text windows, but the need for advanced functions like the *grammar spreadsheet* no longer seem important enough to justify their further development. Dictionaries, thesauri and encyclopedias have become available for free online, and Google can often be useful to check a word's spelling. Translation and summarization systems are also available online.

While the original target for the early dedicated proofreading systems had disappeared, the interest in the relation between NLP and the writing process remained strong and was explored in different ways. Experience with

CRITIQUE had already revealed that different groups profit differently: non-professional writers reported that more than 80% of CRITIQUE's suggestions to them were correct or useful, against 41% of professional writers. Domeij (1998) conducted a study and found that such tools can have a positive effect, but different writers cope differently with these tools. On the one hand, studies like these emphasize the importance of a thorough evaluation of NLP tools for writing in practical use. On the other hand, the larger cognitive and societal context in which writing takes place means that we must also consider the promotion of writing ability in the context of language learning and teaching and in relation to language policy issues.

Language learning and teaching started to become a target for NLP for writing relatively early. Research in proofreading had soon emphasized the distinction between mechanical and cognitive errors. Since the latter are in an obvious relation to language ability as the result of learning, they can be the target of various learning and teaching schemes. On the one hand, second language learners with gaps in their knowledge of the language may benefit not only from corrections but also from additional explanatory material that comes with good proofreading systems. On the other hand, native language learners are sometimes insufficiently aware of homophones with different spellings in different grammatical contexts, e.g. Norwegian *å* vs. *og* or French verbal forms ending in *-er*, *-ez*, or *-é*.

In the early 1990s, the Dutch company Cognitech developed several systems for spelling and grammar learning. Among these, SPELRAAM focused on spelling, and especially homophones, in syntactic contexts. The system is targeted at native speakers of Dutch and uses a decision tree to make learners aware of the grammatical choices that influence a word form. Figures 7–9 are screenshots of this system.

More recently, dedicated writing tools for second language learners were developed that combine proofreading with targeted pedagogical components. The Grim system (Knutsson, 2005) is a prime example of this line of research. By targeting the system to a specific audience, it is easier to optimize its usefulness. This presupposes empirical studies of writing processes and problems. As more data is becoming available, a systematic study of spelling and grammar problems in authentic writing situations is becoming feasible. The ASK project (Tenfjord et al., 2006) has collected a large number of Norwegian essays by students of Norwegian as a second language. These have been carefully error-coded and made searchable. Figure 10 shows a selection of the corpus revealing adjective form errors, while figure 11 shows the different distribution of some error types among different learner groups.

The second link concerns language policy, especially for languages that have complicated spelling systems. Public bodies governing language policy tend to be very interested in promoting good spelling practice among language users. It is interesting that in the preparations for the Dutch spelling reform in the 1990s, consideration was given to NLP applications that would handle this spelling. Ultimately a simplification was achieved by establishing a single official spelling for each word, replacing preferred and less preferred variants. The even more complicated variation in Norwegian presented a headache for SCARRIE. Eventually, the Norwegian partners in SCARRIE solved this by establishing a limited set of subnorms and enabling adherence to a chosen subnorm though sophisticated dictionary and grammar codings. In the wake of this research, attention was drawn to the complications of the subnorms and the fact that many allowed lexical variants do not appear to be ever used (Rosén, 2000). A simplification of the variation in Bokmål was adopted by Norsk Språkråd in 2005 and there are plans for further empirical investigations of the situation. It should also be mentioned that political priorities have spurred the development of special writing tools to promote the participation of people with language-related disorders in social communication. In Norway, companies like Include and LingIT have been active in the development of such tools.

In conclusion, I would like to observe, firstly, that NLP for writing has been a research field that has seen important shifts in its intended application environments during the past couple of decades. Secondly, there are links between NLP for writing and other fields that directly or indirectly benefit from this research or vice versa, including language learning and teaching and language policy. Finally, a holistic approach to writing is needed, where NLP research better interacts with the study of cognitive aspects of the writing process (including first and second language learning and language disorders) and with an investigation of the changing environments for written communication and our appreciation of correctly written texts also in the new media.

References

Domeij, Rickard. 1998. Detecting, diagnosing and correcting low-level problems when editing with and without computer aids. *Text Technology* 8(1):12–25.

Domeij, Rickard, Ola Knutsson, Johan Carlberger, and Viggo Kann. 2000. Granska: An efficient hybrid system for Swedish grammar checking. In *Proceedings of the 12th Nordic Conference on Computational Linguistics (NoDaLiDa)*.

Heidorn, George E., Karen Jensen, Lance Miller, Roy Byrd, and Martin Chodorow. 1982. The EPISTLE text-critiquing system. *IBM Systems Journal* 21(3):305.

Knutsson, Ola. 2005. *Developing and Evaluating Language Tools for Writers and Learners of Swedish*. Ph.D. thesis, Kungliga Tekniska högskolan.

Macdonald, Nina H., L. T. Frase, P. Gingrich, and S. A. Keenan. 1982. The Writer's Workbench: Computer aids for text analysis. In *IEEE Transactions on Communication (Special Issue on Communication in the Automated Office)*, vol. 30, page 105.

Richardson, Stephen D. and Lisa C. Braden-Harder. 1988. The experience of developing a large-scale natural language text processing system: CRITIQUE. In *Proceedings of the 2nd Conference on Applied Natural Language Processing, Austin, TX, 9–12 February 1988*, pages 195–202.

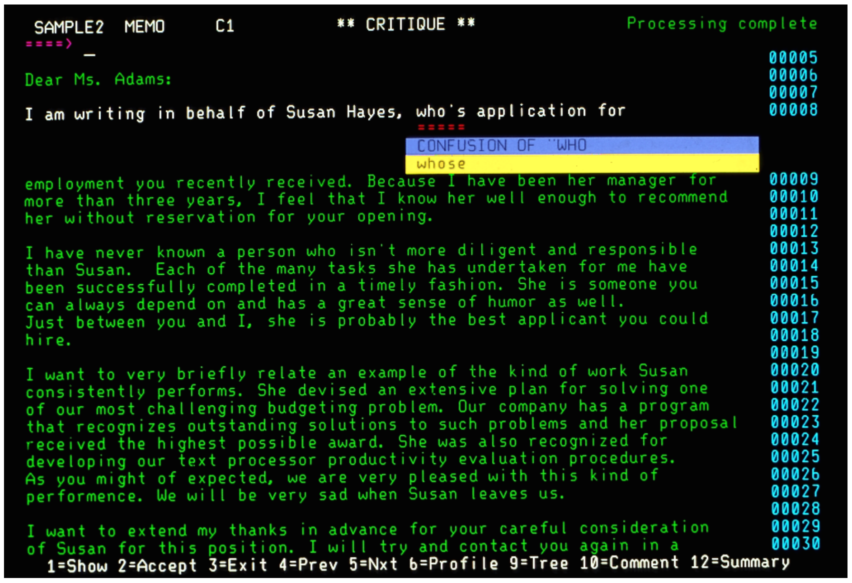
Rosén, Victoria. 2000. Er norsk et naturlig språk? In Ø. Andersen, K. Fløttum, and T. Kinn, eds., *Menneske, språk og felleskap*, pages 157–173. Novus forlag.

Tenfjord, Kari, Paul Meurer, and Knut Hofland. 2006. The ASK corpus: A language learner corpus of Norwegian as a second language. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 1821–1824.

van Berkel, Brigit and Koenraad De Smedt. 1988. Triphone analysis: A combined method for the correction of typographical and orthographical errors. In *Proceedings of the 2nd Conference on Applied Natural Language Processing, Austin, TX, 1988*, pages 77–83. ACL.

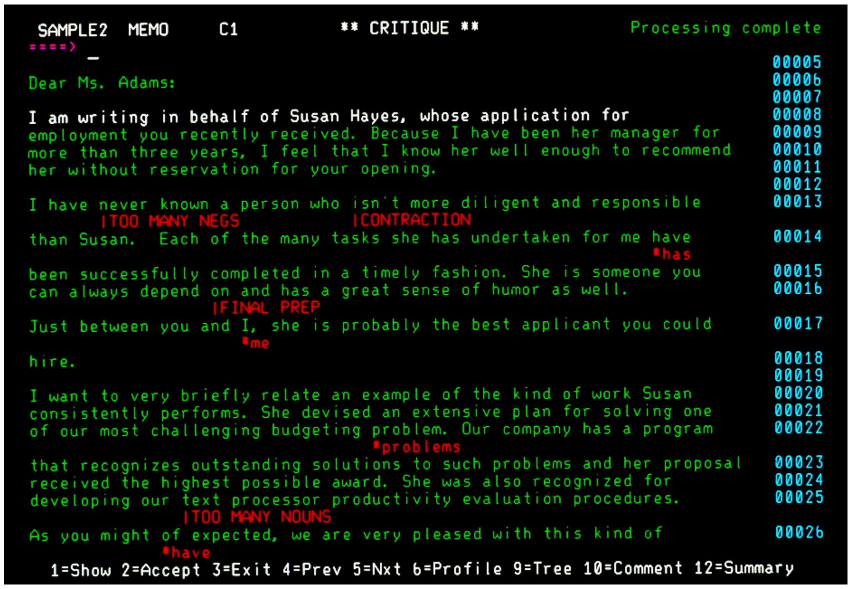
Vosse, Theo. 1994. *The Word Connection: Grammar-based Spelling Error Correction in Dutch*. Enschede: Neslia Paniculata.

Figures



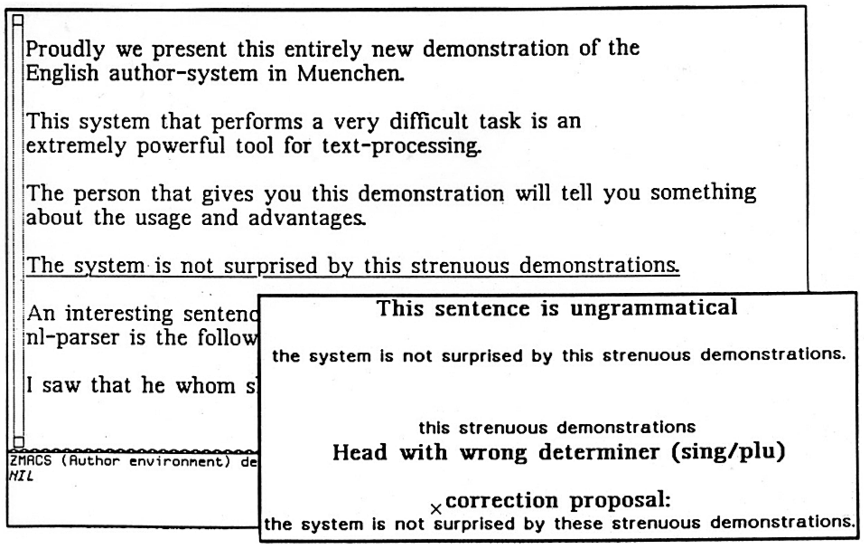
```
 SAMPLE2  MEMO     C1            ** CRITIQUE **          Processing complete
====> _

Dear Ms. Adams:

I am writing in behalf of Susan Hayes, who's application for
                                       =====
                                      CONFUSION OF "WHO
                                      whose
employment you recently received. Because I have been her manager for
more than three years, I feel that I know her well enough to recommend
her without reservation for your opening.

I have never known a person who isn't more diligent and responsible
than Susan.  Each of the many tasks she has undertaken for me have
been successfully completed in a timely fashion. She is someone you
can always depend on and has a great sense of humor as well.
Just between you and I, she is probably the best applicant you could
hire.

I want to very briefly relate an example of the kind of work Susan
consistently performs. She devised an extensive plan for solving one
of our most challenging budgeting problem. Our company has a program
that recognizes outstanding solutions to such problems and her proposal
received the highest possible award. She was also recognized for
developing our text processor productivity evaluation procedures.
As you might of expected, we are very pleased with this kind of
performence. We will be very sad when Susan leaves us.

I want to extend my thanks in advance for your careful consideration
of Susan for this position. I will try and contact you again in a
  1=Show 2=Accept 3=Exit 4=Prev 5=Nxt 6=Profile 9=Tree 10=Comment 12=Summary
```

Figure 1: Screenshot of CRITIQUE proposing a correction (from a 35mm slide courtesy of Stephen Richardson).



```
 SAMPLE2  MEMO     C1           ** CRITIQUE **              Processing complete
====> _

Dear Ms. Adams:

I am writing in behalf of Susan Hayes, whose application for
employment you recently received. Because I have been her manager for
more than three years, I feel that I know her well enough to recommend
her without reservation for your opening.

I have never known a person who isn't more diligent and responsible
       |TOO MANY NEGS           |CONTRACTION
than Susan.  Each of the many tasks she has undertaken for me have
                                                             *has
been successfully completed in a timely fashion. She is someone you
can always depend on and has a great sense of humor as well.
                  |FINAL PREP
Just between you and I, she is probably the best applicant you could
                      *me
hire.

I want to very briefly relate an example of the kind of work Susan
consistently performs. She devised an extensive plan for solving one
of our most challenging budgeting problem. Our company has a program
                                    *problems
that recognizes outstanding solutions to such problems and her proposal
received the highest possible award. She was also recognized for
developing our text processor productivity evaluation procedures.
               |TOO MANY NOUNS
As you might of expected, we are very pleased with this kind of
             *have
  1=Show 2=Accept 3=Exit 4=Prev 5=Nxt 6=Profile 9=Tree 10=Comment 12=Summary
```

Figure 2: Screenshot of CRITIQUE highlighting suspected errors (from a 35mm slide courtesy of Stephen Richardson).



Proudly we present this entirely new demonstration of the English author-system in Muenchen.

This system that performs a very difficult task is an extremely powerful tool for text-processing.

The person that gives you this demonstration will tell you something about the usage and advantages.

The system is not surprised by this strenuous demonstrations.

An interesting sentenc
nl-parser is the follow

I saw that he whom s

ZMACS (Author environment) de
NIL

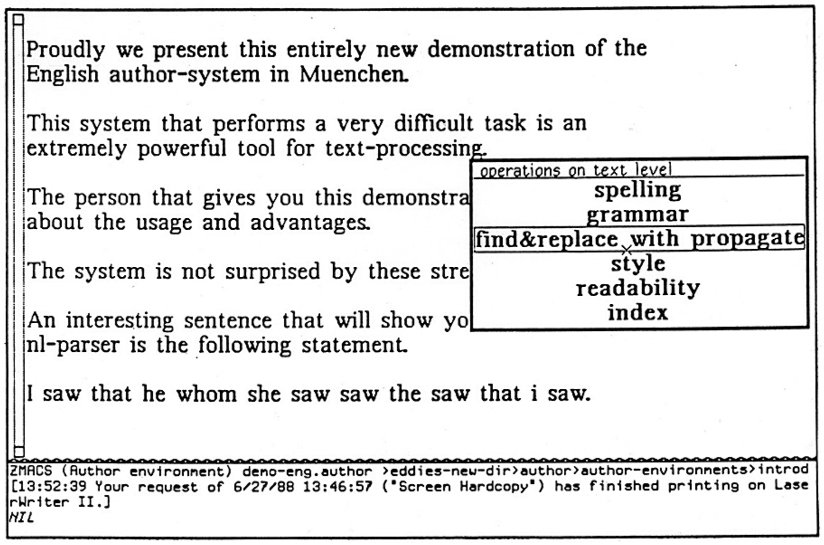
This sentence is ungrammatical

the system is not surprised by this strenuous demonstrations.

this strenuous demonstrations
Head with wrong determiner (sing/plu)

correction proposal:
the system is not surprised by these strenuous demonstrations.

Figure 3: Screenshot of Author Environment proposing a diagnosis and correction.



Proudly we present this entirely new demonstration of the
English author-system in Muenchen.

This system that performs a very difficult task is an
extremely powerful tool for text-processing.

The person that gives you this demonstra
about the usage and advantages.

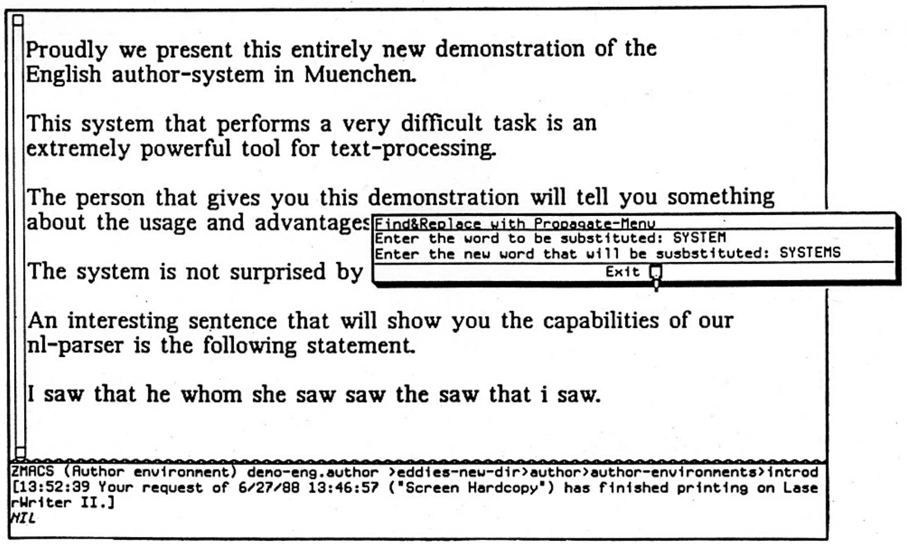
The system is not surprised by these stre

An interesting sentence that will show yo
nl-parser is the following statement.

I saw that he whom she saw saw the saw that i saw.

```
ZMACS (Author environment) demo-eng.author >eddies-new-dir>author>author-environments>introd
[13:52:39 Your request of 6/27/88 13:46:57 ("Screen Hardcopy") has finished printing on Lase
rWriter II.]
NIL
```

Figure 4: Screenshot of Author Environment menu including 'Find and replace with propagate'.



Proudly we present this entirely new demonstration of the
English author-system in Muenchen.

This system that performs a very difficult task is an
extremely powerful tool for text-processing.

The person that gives you this demonstration will tell you something
about the usage and advantages

The system is not surprised by

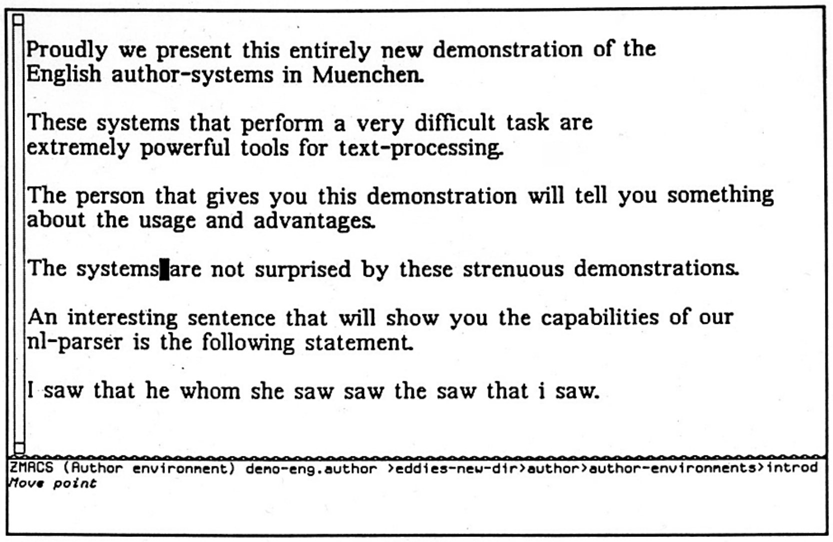
An interesting sentence that will show you the capabilities of our
nl-parser is the following statement.

I saw that he whom she saw saw the saw that i saw.

```
Find&Replace with Propagate-Menu
Enter the word to be substituted: SYSTEM
Enter the new word that will be susbstituted: SYSTEMS
Exit
```

```
ZMACS (Author environment) demo-eng.author >eddies-new-dir>author>author-environments>introd
[13:52:39 Your request of 6/27/88 13:46:57 ("Screen Hardcopy") has finished printing on Lase
rWriter II.]
NIL
```

Figure 5: Screenshot of Author Environment where a word is being replaced.



Proudly we present this entirely new demonstration of the
English author-systems in Muenchen.

These systems that perform a very difficult task are
extremely powerful tools for text-processing.

The person that gives you this demonstration will tell you something
about the usage and advantages.

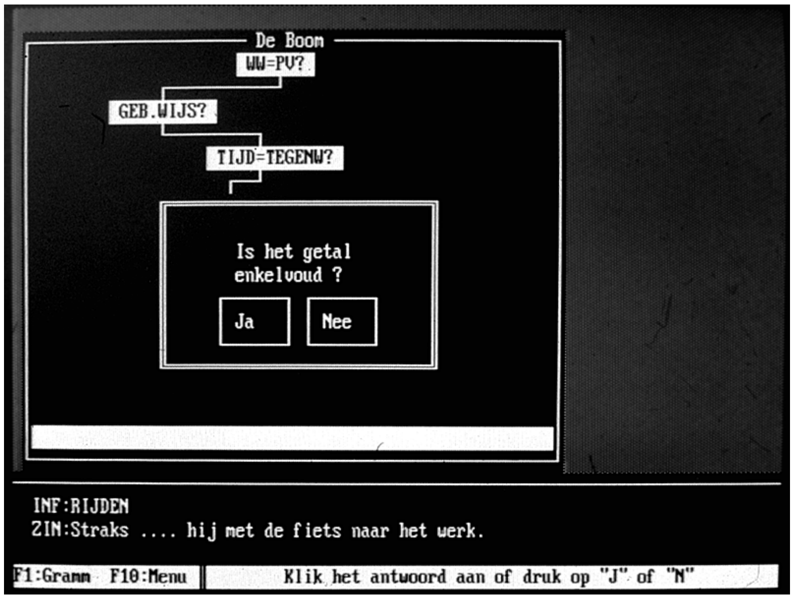
The systems are not surprised by these strenuous demonstrations.

An interesting sentence that will show you the capabilities of our
nl-parser is the following statement.

I saw that he whom she saw saw the saw that i saw.

```
ZMACS (Author environment) demo-eng.author >eddies-new-dir>author>author-environments>introd
Move point
```

Figure 6: Screenshot of Author Environment showing the result of propagating a change from singular to plural.



De Boom

WW=PV?

GEB.WIJS?

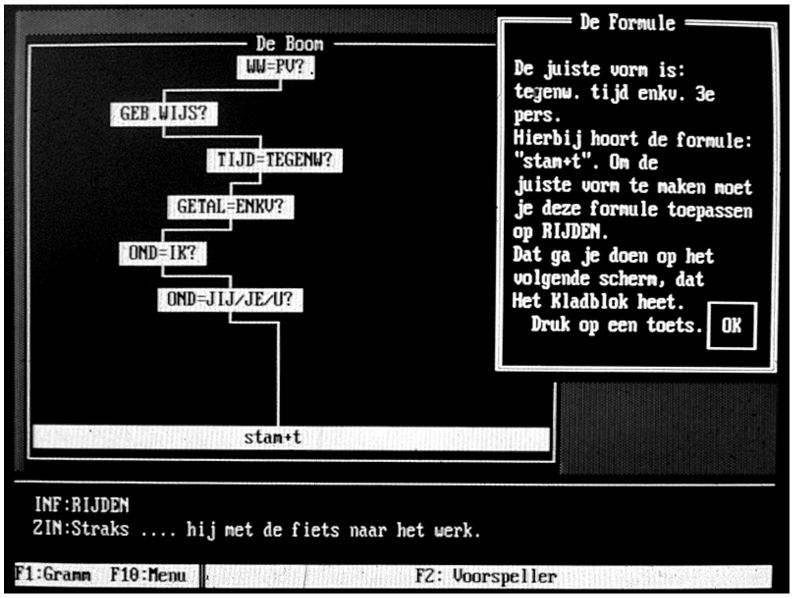
TIJD=TEGENW?

Is het getal
enkelvoud ?

Ja Nee

```
INF:RIJDEN
ZIN:Straks .... hij met de fiets naar het werk.
F1:Gramm  F10:Menu    Klik het antwoord aan of druk op "J" of "N"
```

Figure 7: Screenshot of SPELRAAM showing how the user completes a decision tree (from a 35mm slide courtesy of Gerard Kempen).


De Boom

WW=PV?

GEB.WIJS?

TIJD=TEGENW?

GETAL=ENKV?

OND=IK?

OND=JIJ/JE/U?

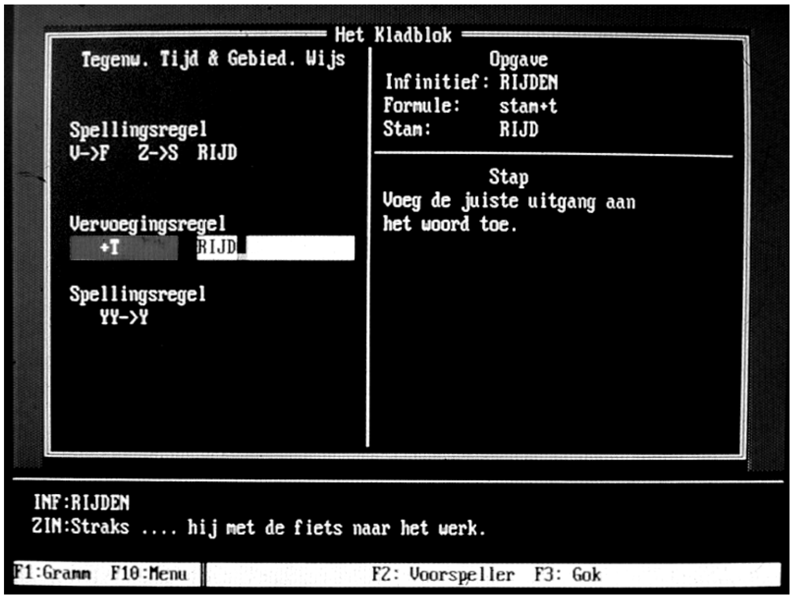
stam+t

De Formule

De juiste vorm is:
tegenw. tijd enkv. 3e
pers.
Hierbij hoort de formule:
"stam+t". Om de
juiste vorm te maken moet
je deze formule toepassen
op RIJDEN.
Dat ga je doen op het
volgende scherm, dat
Het Kladblok heet.
Druk op een toets.

INF:RIJDEN
ZIN:Straks .... hij met de fiets naar het werk.

Figure 8: Screenshot of SPELRAAM showing a spelling rule for conjugation (from a 35mm slide courtesy of Gerard Kempen).


Het Kladblok

Tegenw. Tijd & Gebied. Wijs

Spellingsregel
V->F Z->S RIJD

Vervoegingsregel
+T RIJD

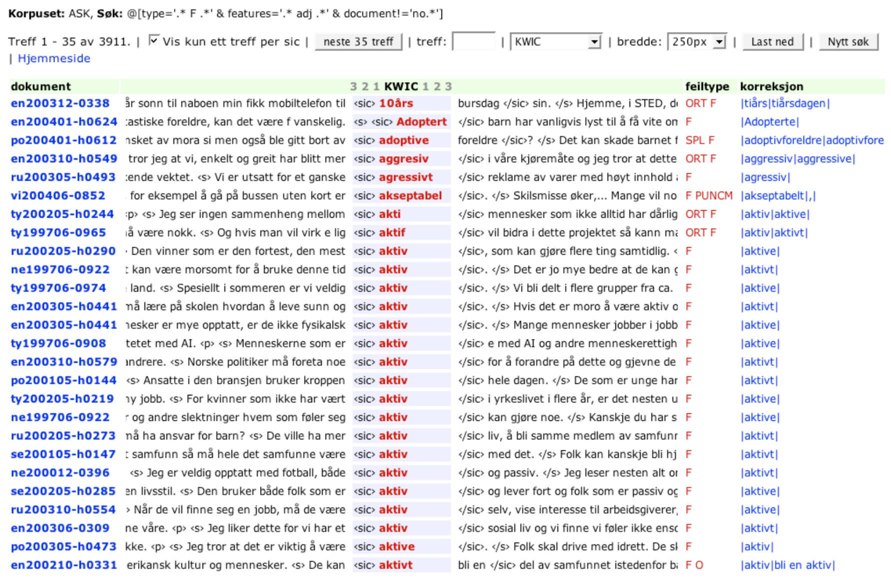
Spellingsregel
YY->Y

Opgave
Infinitief: RIJDEN
Formule: stam+t
Stam: RIJD

Stap
Voeg de juiste uitgang aan
het woord toe.

INF:RIJDEN
ZIN:Straks .... hij met de fiets naar het werk.

Figure 9: Screenshot of SPELRAAM giving spelling advice for conjugation by applying a rule (from a 35mm slide courtesy of Gerard Kempen).

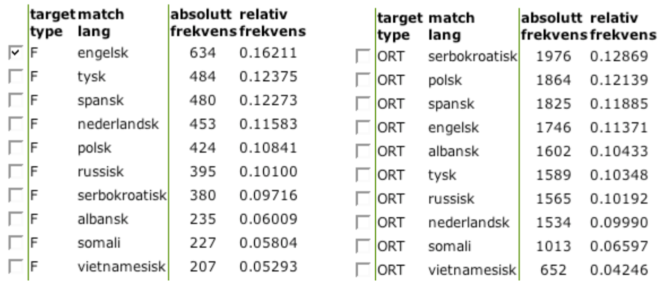


Korpuset: ASK, **Søk:** @[type='.* F .*' & features='.* adj .*' & document!='no.*']

Treff 1 - 35 av 3911. | ✓ Vis kun ett treff per sic | neste 35 treff | treff: | KWIC | bredde: 250px | Last ned | Nytt søk
| Hjemmeside

dokument		3 2 1 KWIC 1 2 3		feiltype	korreksjon
en200312-0338	år sonn til naboen min fikk mobiltelefon til	<sic> 10års	bursdag </sic> sin. </s> Hjemme, i STED, de	ORT F	|tiårs|tiårsdagen|
en200401-h0624	:astiske foreldre, kan det være f vanskelig.	<s> <sic> Adoptert	</sic> barn har vanligvis lyst til å få vite om	F	|Adopterte|
po200401-h0612	nsket av mora si men også ble gitt bort av	<sic> adoptive	foreldre </sic>? </s> Det kan skade barnet f	SPL F	|adoptivforeldre|adoptivfore
en200310-h0549	tror jeg at vi, enkelt og greit har blitt mer	<sic> aggresiv	</sic> i våre kjøremåte og jeg tror at dette	ORT F	|aggressiv|aggressive|
ru200305-h0493	:ende vektet. <s> Vi er utsatt for et ganske	<sic> agressivt	</sic> reklame av varer med høyt innhold a	F	|agressiv|
vi200406-0852	for eksempel å gå på bussen uten kort er	<sic> akseptabel	</sic>. </s> Skilsmisse øker,... Mange vil no	F PUNCM	|akseptabelt|,|
ty200205-h0244	:p> <s> Jeg ser ingen sammenheng mellom	<sic> akti	</sic> mennesker som ikke alltid har dårlig	ORT F	|aktiv|aktive|
ty199706-0965	ıå være nokk. <s> Og hvis man vil virk e lig	<sic> aktif	</sic> vil bidra i dette projektet så kann ma	ORT F	|aktiv|aktivt|
ru200205-h0290	· Den vinner som er den fortest, den mest	<sic> aktiv	</sic>, som kan gjøre flere ting samtidlig. <	F	|aktive|
ne199706-0922	: kan være morsomt for å bruke denne tid	<sic> aktiv	</sic>. </s> Det er jo mye bedre at de kan g	F	|aktivt|
ty199706-0974	: land. <s> Spesiellt i sommeren er vi veldig	<sic> aktiv	</sic>. </s> Vi bli delt i flere grupper fra ca.	F	|aktive|
en200305-h0441	nå lære på skolen hvordan å leve sunn og	<sic> aktiv	</sic>. </s> Hvis det er moro å være aktiv o	F	|aktivt|
en200305-h0441	ıesker er mye opptatt, er de ikke fysikalsk	<sic> aktiv	</sic>. </s> Mange mennesker jobber i jobb	F	|aktive|
ty199706-0908	tetet med AI. <p> <s> Menneskerne som er	<sic> aktiv	</sic> e med AI og andre menneskerettigh	F	|aktive|
en200310-h0579	ındrere. <s> Norske politiker må foreta noe	<sic> aktiv	</sic> for å forandre på dette og gjevne de	F	|aktivt|
po200105-h0144	<s> Ansatte i den bransjen bruker kroppen	<sic> aktiv	</sic> hele dagen. </s> De som er unge har	F	|aktivt|
ty200205-h0219	ıy jobb. <s> For kvinner som ikke har vært	<sic> aktiv	</sic> i yrkeslivet i flere år, er det nesten u	F	|aktive|
ne199706-0922	r og andre slektninger hvem som føler seg	<sic> aktiv	</sic> kan gjøre noe. </s> Kanskje du har s	F	|aktive|
ru200205-h0273	nå ha ansvar for barn? <s> De ville ha mer	<sic> aktiv	</sic> liv, å bli samme medlem av samfunr	F	|aktivt|
se200105-h0147	: samfunn så må hele det samfunne være	<sic> aktiv	</sic> med det. </s> Folk kan kanskje bli hj	F	|aktivt|
ne200012-0396	<s> Jeg er veldig opptatt med fotball, både	<sic> aktiv	</sic> og passiv. </s> Jeg leser nesten alt oı	F	|aktivt|
se200205-h0285	en livsstil. <s> Den bruker både folk som er	<sic> aktiv	</sic> og lever fort og folk som er passiv og	F	|aktive|
ru200310-h0554	› Når de vil finne seg en jobb, må de være	<sic> aktiv	</sic> selv, vise interesse til arbeidsgiverer,	F	|aktive|
en200306-0309	ne våre. <p> <s> Jeg liker dette for vi har et	<sic> aktiv	</sic> sosial liv og vi finne vi føler ikke ensc	F	|aktivt|
po200305-h0473	kke. <p> <s> Jeg tror at det er viktig å være	<sic> aktive	</sic>. </s> Folk skal drive med idrett. De sl	F	|aktiv|
en200210-h0331	erikansk kultur og mennesker. <s> De kan	<sic> aktivt	bli en </sic> del av samfunnet istedenfor bi	F O	|aktiv|bli en aktiv|

Figure 10: Screenshot of KWIC search result for wrong forms of adjectives in ASK.



	target type	match lang	absolutt frekvens	relativ frekvens
✓	F	engelsk	634	0.16211
☐	F	tysk	484	0.12375
☐	F	spansk	480	0.12273
☐	F	nederlandsk	453	0.11583
☐	F	polsk	424	0.10841
☐	F	russisk	395	0.10100
☐	F	serbokroatisk	380	0.09716
☐	F	albansk	235	0.06009
☐	F	somali	227	0.05804
☐	F	vietnamesisk	207	0.05293

	target type	match lang	absolutt frekvens	relativ frekvens
☐	ORT	serbokroatisk	1976	0.12869
☐	ORT	polsk	1864	0.12139
☐	ORT	spansk	1825	0.11885
☐	ORT	engelsk	1746	0.11371
☐	ORT	albansk	1602	0.10433
☐	ORT	tysk	1589	0.10348
☐	ORT	russisk	1565	0.10192
☐	ORT	nederlandsk	1534	0.09990
☐	ORT	somali	1013	0.06597
☐	ORT	vietnamesisk	652	0.04246

Figure 11: Frequencies of two error types in ASK, grouped according to mother tongue: Wrong form of adjective (left); orthographical error (right).

Keystroke logging – a didactic tool for analysis and development of writing and language skills

Eva Lindgren and **Kirk Sullivan**
Umeå University
Sweden
eva.lindgren@educ.umu.se, kirk@ling.umu.se

Abstract

This paper presents several studies in which keystroke logging has been used as a didactic tool for writing and language development. Keystroke logging, as presented here, provides learners with a tool for analysis and reflection on their own written production and teachers with a tool for analysis and individual feed-back. The paper aims to outlines the theoretical assumptions behind the studies, discuss the impact of keystroke logging on writing and language development and critically examine the results from a classroom perspective.

1. Keystroke logging

Keystroke logging is a method that is represented though a number of software programmes (e.g. JEdit, Scriptlog, Inputlog, Translog) which all share the basic common principles of recording every keystroke and mouse action a writer undertakes during a writing session. The programmes typically include a replay function and various statistics about, for example, pauses and revisions. The data enables export to other tools for visualisation and statistical analyses.

1.1 An awareness-raising tool

When used retrospectively, the replay function provides writers with an opportunity to observe their own writing process in detail. The main advantages of such an approach are 1) that learners' cognitive load is reduced, 2) that noticing is promoted, and 3) that learners are provided with input on a suitable level. A group of young writers improved their texts in their first language after such a reflection and discussion session (Lindgren, 2005). In their foreign language retrospective replay and discussion enhanced their awareness of, in particular, stylistic aspects of writing and the reader (Lindgren, Stevenson and Sullivan, 2008).

1.2 A tool for analysis

For instructors, keystroke logging provides a tool for analysis. Through the automatic analyses of pauses and revisions, measures of fluency can be easily calculated (Spelman-Miller, Lindgren and Sullivan, 2008; Lindgren, Spelman-Miller and Sullivan, in press). Measures of fluency include the length of text span writers produce between interruption, i.e. a pause or a revision. By measuring fluency instructors receive indications of writers development in a first or a foreign language. Higher fluency indicates that a writer has achieved a higher level of automatisation of writing or language aspects, such as spelling. Higher level of automatisation of spelling enables writers to focus more on other aspects of writing, which results in better text quality. Further, the automatic data can be used for visualisation, which can assist both learners, individually or class, and instructors in understanding what goes on during writing (Lindgren and Sullivan, 2002; Lindgren, Spelman-Miller, Lindgren and Sullivan, 2007).

2. Conclusions

The studies above present positive results of the use of keystroke logging as a didactic tool. However, they also raise questions of how to best use the method to maximise the result for each individual writer, how to best provide feed-back and whether the method is useful for all learners.

References

Lindgren, E. (2005). The uptake of peer-based intervention in the writing classroom Rijlaarsdam, G., Van den Bergh, H. & Couzijn, M. (Vol. Eds.), Studies in writing, Volume 14, Effective learning and teaching of writing, 2nd edition, (259–274). Dordrecht: Kluwer Academic Publishers.

Lindgren, E. and Sullivan K.P.H. (2002). The LS graph: A methodology for visualising writing revision. Language Learning 52(3), 565–595.

Lindgren, E., Sullivan, K.P.H., & Spelman Miller, K. (in press, 2008). Development of fluency and revision in L and L2 writing in Swedish high school years 8 and 9. *International Journal of Applied Linguistics.*

Spelman Miller, K., Lindgren, E., & Sullivan, K.P.H. (2008). The psycholinguistic dimension in second language writing: opportunities for research and pedagogy. *TESOL Quarterly*.

Lindgren, E., Sullivan, K.P.H. & Stevenson, M. (2008). Supporting the reflective language learner with computer keystroke logging. In B. Barber and F. Zhang (Eds.), *Handbook of Research on Computer Enhanced Language Acquisition and Learning* (pp. 189 – 204). Hershey, NY: Information Science Reference, IGI Global.

Lindgren, E., Sullivan, K.P.H., Lindgren, U & Spelman Miller, K. (2007). GIS for writing: applying geographic information system techniques to data-mine writing's cognitive processes. In G. Rijlaarsdam (Series Ed.) and M. Torrance, L. Van Waes & D. Galbraith (Vol. Eds), *Writing and Cognition: Research and Applications* (pp. 83–96). Amsterdam: Elsevier.

Opportunities and Limits for Language Awareness in Text Editors

Cerstin Mahlow and **Michael Piotrowski**
Institute of Computational Linguistics
University of Zurich
Zurich, Switzerland
{mahlow, mxp}@cl.uzh.ch

Abstract

In this paper we argue that the concept of *language awareness*, as known from programmer's editors, can be transferred to writing natural language and word processors. We propose editing functions which use methods from computational linguistics and take the structures of natural languages into consideration. Such functions could reduce errors and better support writers in realizing their communicative goals. We briefly compare characteristics of programming languages and natural languages and their processing tools with respect to their suitability for being used in language-aware functions in editors. However, linguistic methods have limits, and there are various aspects software developers have to take into account to avoid creating a solution looking for a problem: Language-aware functions could be powerful tools for writers, but writers must not be forced to adapt to their tools.

1 Introduction

Writing is a daily task for a great number of people. However, today's word processors offer only limited support for writing and editing: Most functions are character-based and thus force writers to translate high-level goals into low-level functions of the editor. This causes typical errors, e.g., missing verbs, agreement errors, or wrong word order. Functions improving the "brain-to-hand-to-keyboard-to-screen-connection" (Taylor, 1987, p. 79) as proposed by Dale (1989; 1996) or Mahlow and Piotrowski (2008) could help avoid several types of errors. Additionally, as cognitive resources are limited (McCutchen, 1996; Allen and Scerbo, 1983), language-aware functions could reduce the effort needed to deal with word processors and help writers concentrate on their actual goals and remain in control of their text. Writers should get interactive support during writing, very similar to the support programmers get from their editors during programming.

We will first describe the principles of language awareness as they can be deduced from respective functions in programmer's editors and have a look at the current situation in word processors. Then we propose interactive editing functions operating on linguistic elements and making use of tools and methods of computational linguistics. From the comparison of characteristics of programming languages and natural languages and the hence resulting quality of the respective language processing tools we will deduce opportunities and limits language technologists have to be aware of when implementing language-aware functions for word processors.

2 Language Awareness in Editors

Both text written in natural languages (such as English, German, or French) and computer programs written in programming languages (such as Perl, Python, or C) have underlying syntactic structures and are not merely strings of characters. In the context of programming languages, text editors which are aware of this structure and use this awareness to support the creation and editing of programs are referred to as *syntax-directed*, *language-sensitive*, *language-based*, or *language-aware editors* (Khwaja and Urban, 1993).

The ultimate goal of language awareness in programmer's editors is to *prevent errors* in programs, as the prevention of errors helps producing higher-quality programs. Language awareness supports programmers by giving them a better overview of programs and by providing them with editing functions operating on structural elements instead of characters or lines.

In general, we can distinguish two types of language-aware functionality: (1) *Information functions* for highlighting individual language elements and larger structures, or for displaying statistical information regarding certain elements, which do not change the text, and (2) *operations* for inserting, reordering, modifying, or deleting elements, i.e., functions changing the text. Both types of functions operate on the elements defined by the lexicon and the morphological and syntactical rules of a concrete language.

Just as we can distinguish between formal languages (in this case: programming languages) and natural languages, we can distinguish between two major types of editors: Editors intended for writing computer programs (*programmer's editors*) and editors intended for writing natural-language text (usually referred to as *word processors*). These two types of editors can be seen as two instances of the general class of text editors, each adapted to handle writing, editing, and revising in specific languages.

We will now briefly analyze language awareness in these two types of editors.

2.1 Language Awareness in Programmer's Editors

Programmer's editors generally implement both information functions and operations. Many editors support different programming languages through language-specific editing *modes*, which are either activated automatically or can be selected by the user.

Syntax highlighting is the most prominent instantiation of an information function: Keywords, variable names, and specific constructs can be highlighted using different colors or fonts. Programmer's editors generally also help to ensure that parentheses are properly nested, e.g., by highlighting mismatches.

As instantiations of language-aware operations we can typically find functions for deleting elements, e.g., parenthesized expressions or comments, for inserting or completing syntactic structures, such as conditional expressions or looping constructs, and for selecting certain syntactic elements (e.g., the current function definition) for a subsequent operation. Some editors offer code completion, i.e., the editor can complete an initial string typed by the user, either automatically or upon request. The editor may take the context into account; for example, in an object-oriented programming language it may consider only the names of those methods available for the particular object.

Programmer's editors also indent lines automatically according to the syntax and may control the insertion of whitespace and newlines (e.g., around operators or after block-opening braces). In some languages, such as Python, indentation serves to indicate the block structure of the code. For languages like Perl, C, Java, or Lisp, indentation is not mandatory but conventionally reflects the syntactic structure, which is also marked by parentheses or braces.

2.2 Language Awareness in Word Processors

Since programmer's editors support developers with specific functions for the programming language being used, word processors could be expected to offer specific functions depending on the language the writer is using.

However, unlike programmer's editors, word processors offer very few language-aware functions: Almost all functions are based on characters and lines. Thus, even state-of-the-art word processors offer only a basic set of core operations (e.g., select, cut, copy, paste, insert) (Piolat, 1991, p. 262), (Sharples and Pemberton, 1990, p. 49), regardless of the language the writer is using.

Checkers for spelling, grammar, and style, which are nowadays available for various languages in many word processors, provide a certain level of "language awareness." However, regardless of their quality (Vernon, 2000; McGee and Ericsson, 2002), they are essentially tools for *post-writing*: After a draft is finished, they can detect errors and propose modifications, but they generally do not support writers *during* writing and editing, and thus do not help to *prevent* errors.

This situation clearly is disappointing. Considering the fact that there already exist sophisticated natural-language-processing methods and tools, we think the time has come to add language-aware functions to word processors as well.

3 Language-Aware Editing Functions in Word Processors

Writers should receive interactive support from their word processors, similar to the interactive support programmers get. Supporting writers during the writing and editing process reduces the cognitive load and therefore helps avoiding errors.

Writers should be in control of their text, relying on post-processing support only denies the fact that writing is a very active and creative process.

We propose two types of functions operating on linguistic elements, such as words, phrases, or clauses. These functions are intended to work analogously to the corresponding functions known for programmer's editors: (1) *Information functions* for highlighting elements, such as verbs or PP-attachments, or for providing writers with information about certain aspects of the text, such as prepositions used, sentences without verbs, or variants of multi-word expressions. Writers can interpret the results themselves and decide how to make use of them. (2) *Operations* for reordering, modifying, or deleting linguistic elements. In order to reduce the cognitive load, the number of actions necessary to reach a specific goal should be reduced drastically by combining sequences of core operations into higher-level functions closer to writers' goals and their mental model of the task. Examples would be the pluralization of an entire phrase (a complex task for morphologically rich languages as German), the reordering of conjunctions, or the replacing of words or phrases through the whole text (also a complex task for highly inflectional languages). See Mahlow et al. (2008) for more details.

Both types of functions require *linguistic knowledge* and *linguistic resources*. Linguistic knowledge will influence the ideal combination of existing core operations into higher-level functions a user can call with one keystroke: Reordering conjuncts is a highly complex task if a writer has to find the sequence of core operations on their own; using *one* operation reduces the risk of producing ungrammatical conjuncts. Linguistic resources will be needed for operations that modify certain linguistic elements: Pluralization of entire phrases will obviously require morphological analysis and generation.

4 Natural and Programming Languages

It is clear that there are significant differences between programming languages and natural languages. Two important differences are:

1. The lexicon: The lexicon of programming language is small and essentially closed. The lexicon for a natural language is much bigger and can be extended *ad infinitum* by morphological processes.
2. The syntactical rules: Syntactical rules for formal languages are made *a priori*, i.e., prior of creating a language. Users are not allowed to change the rules. Syntactical rules for a natural language, however, try to describe the phenomena of a certain language *a posteriori*. Natural languages "live," i.e., users change the rules as they are using the language – generally, native speakers are not even aware of the rules. Linguists can only discover and adapt the rules of a language *afterwards* by observing the language.

Thus, as the lexicon of programming languages is relatively small and – most importantly – closed, functions for highlighting keywords, can be implemented relatively easily. There are strict rules for extending the "lexicon" of a language with variable names (e.g., a name for a hash variable in Perl has to begin with a "%"), so that these can generally also be detected easily.

These properties of programming and natural languages explain the difference in performance of parsers for the respective types of languages: Processors of programming languages can be implemented easily, they are very sophisticated, work very fast and deliver satisfying and reliable results. Processors of natural language struggle with incomplete rules, ambiguities, big and always incomplete resources, their results in general are not very convincing, they need much time to deliver these results, thus making them not very attractive for interactive use – it is not acceptable for a writer to wait several seconds for a phrase to be pluralized.

However, there exist morphological and syntactical parsers for several natural languages which work quite satisfactorily for restricted phenomena or purposes. Additionally, nowadays computers have sufficient processing power to reduce the time needed to analyze word forms or generate phrases drastically compared to the situation ten years ago. We therefore propose to make use of those processing tools in word processors in a similar way programmer's editors make use of parsers for programming languages.

5 Opportunities and Limits

5.1 Opportunities

Language-aware operations using syntactical and morphological components could offer writers new ways of working creatively with their texts: With one click they could apply changes to their texts, inspect the results, undo them, and try a different change. They could concentrate on their goal, play with words and phrases, and would not have to care about how to realize these changes, would not have to worry about forgetting one occurrence, and would not have to keep in mind that other locations may need changes because of the original change (e.g., pluralizing the subject of a sentence requires adjustment of the finite verb).

Like syntax highlighting and indentation in programmer's editors assists programmers, the highlighting of specific linguistic elements could help writers to get a better overview of the structure of their text written so far or to identify characteristics with respect to style, e.g., overuse of certain conjunctions or identical beginnings of sentences. When linguistic resources are carefully chosen and cleverly combined with the existing core functionality of word processors, and when the principles of the respective language are taken into account and the available computing power is utilized, various interesting scenarios for language-aware functionality emerge (see Mahlow and Piotrowski (2008) and Mahlow et al. (2008)).

5.2 Limits

While today's computers are capable of performing analyses and generation of linguistic structures fast enough to be suitable for interactive use, linguistic components usually fail to produce results that are 100% correct in terms of precision and recall. Furthermore, for most of these components it cannot be predicted whether the results will be correct. When using them as basis for language-aware functions in word processors, writers must be aware that they should not blindly trust the system to avoid frustrations similar to those often associated with checkers.

A second limit are cases where linguistic resources can deliver correct, but ambiguous results, e.g., it may not be possible to determine the exact category of a word form. The editing function then cannot be executed automatically but has to interact with the writer to resolve the ambiguity. For example, the plural of the German word *Mutter* 'mother; screw nut' may either be *Mütter* or *Muttern*, depending on which of the two meanings are intended.

A third limit is the danger of concentrating on aspects of (computational) linguistics rather than on aspects of the writing process and on writers' needs. For example, at first glance, it seems to be obvious that only operations resulting in grammatically well-formed structures should be allowed. But, on the one hand, the structure may not (yet) be completed and therefore not well-formed *before* executing an operation (e.g., when pluralizing a phrase consisting only of a determiner and an adjective, and the noun is added only after pluralizing). On the other hand, the relevant operation may be used only as one step in a complex sequence: After executing this operation more changes will be applied, and the result is not the end result (e.g., a list of word forms, clearly not a phrase, shall be pluralized, and some of these are then moved to other parts of the text).

This has also been realized during the development of programmer's editors: Especially in the 1980s and 1990s there have been many attempts at programmer's editors which are not based on characters and lines at all but where the programmer instead edits the abstract syntax tree of the program directly (Khwaja and Urban, 1993), thus ensuring that the program was syntactically valid at all times. However, programmers did not accept this type of syntax-directed editors; one important problem was that it also prohibits invalid intermediate states, making editing very cumbersome (Neal, 1987). Current programmer's editors are therefore based on the textual program representation and only provide assistance as described in section 2.1 above.

Syntactic variability may also be considered a problem for language awareness. For languages with free word order, such as German, we can have sentences like:

(1) Ich gab dem Kind gestern einen Apfel.

(2) Gestern gab ich dem Kind einen Apfel.

(3) Dem Kind gab ich gestern einen Apfel.

All syntactical variants express the same basic meaning, 'yesterday I gave an apple to the child.' Another example are passive and active versions of a sentence. Obviously syntactic variants slightly change the meaning of a sentence by changing the focus, but they still express the same main idea. Syntactic variants are one aspect of creativity in writing.

However, similar phenomena also exist in programming languages: Just as in natural languages, one meaning can be expressed in different ways.

In addition, there are also many ways to layout the code, e.g., by using more or less line breaks.

Programmer's editors provide valuable support for programmers without restricting their creativity by forcing them to use one specific syntactic structure for expressing something. This would in fact be impossible since the editor is not able to predict what the programmer has in mind. The same applies to natural-language editing.

6 Conclusion

We have presented the concept of language-aware functions in word processors using methods and systems from computational linguistics. They represent opportunities for supporting the writing process, but developers should avoid concentrating on technical aspects alone, expecting writers to adapt to their tools, which would cause dissatisfaction and ultimately rejection of the tools. The goal clearly must be to support writers by lowering the cognitive effort for complex operations and at the same time allowing them to define *their* goals and to be in control of their texts. This principle has to direct the implementation with respect to technology and usability. Today's state-of-the-art methods and tools for NLP and the available computing power can be used – *and should be used* – to develop language-aware functions for interactive support in word processors.

In the *LingURed* project (see `http://www.lingured.info`) we are developing prototypical implementations of various language-aware editing functions. Depending on licences for the used resources we will publish these functions under an open source licence, and we will evaluate them for usability and effectiveness together with experts in writing research.

References

[Allen and Scerbo1983] Robert B. Allen and M. W. Scerbo. 1983. Details of command-language keystrokes. *ACM Trans. Inf. Syst.*, 1(2):159–178, April.

[Dale and Douglas1996] Robert Dale and Shona Douglas. 1996. Two investigations into intelligent text processing. In Mike Sharples and Thea van der Geest, editors, *The New Writing Environment: Writers at Work in a World of Technology*, chapter 8, pages 123–145. Springer.

[Dale1989] Robert Dale. 1989. Computer-based editorial aids. In Jeremy Peckham, editor, *Recent Developments and Applications of Natural Language Processing*, chapter 2, pages 8–22. Kogan Page Limited.

[Khwaja and Urban1993] Amir A. Khwaja and Joseph E. Urban. 1993. Syntax-directed editing environments: issues and features. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, pages 230–237, New York, NY, USA. ACM.

[Mahlow and Piotrowski2008] Cerstin Mahlow and Michael Piotrowski. 2008. Linguistic support for revising and editing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: 9th International Conference, CICLing 2008, Haifa, Israel, February 17–23, 2008. Proceedings*, pages 631–642, Heidelberg. Springer.

[Mahlow et al.2008] Cerstin Mahlow, Michael Piotrowski, and Michael Hess. 2008. Language-aware text editing. In Robert Dale, Aurélien Max, and Michael Zock, editors, *LREC 2008 Workshop on NLP Resources, Algorithms and Tools for Authoring Aids*, pages 9–13, Marrakech, Morrocco. ELRA.

[McCutchen1996] Deborah McCutchen. 1996. A capacity theory of writing: Working memory in composition. *Educational Psychology Review*, 8(3):299–325.

[McGee and Ericsson2002] Tim McGee and Patricia Ericsson. 2002. The politics of the program: MS Word as the invisible grammarian. *Computers and Composition*, 19(4):453–470, December.

[Neal1987] Lisa R. Neal. 1987. Cognition-sensitive design and user modeling for syntax-directed editors. In *CHI '87: Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, pages 99–102, New York, NY, USA. ACM.

[Piolat1991] Annie Piolat. 1991. Effects of word processing on text revision. *Language and Education*, 5(4):255–272.

[Sharples and Pemberton1990] Mike Sharples and Lyn Pemberton. 1990. Starting from the writer: Guidelines for the design of user-centred document processors. *Computer Assisted Language Learning*, 2(1):37–57.

[Taylor1987] Lee R. Taylor. 1987. Software views: A fistful of word-processing programs. *Computers and Composition*, 5(1):79–90.

[Vernon2000] Alex Vernon. 2000. Computerized grammar checkers 2000: capabilities, limitations, and pedagogical possibilities. *Computers and Composition*, 17(3):329–349, December.

A language technology test bench – automatized testing in the Divvun project

Sjur Nørstebø Moshagen
Norwegian Sámi Parliament
Norway

sjur.moshagen@samediggi.no

Abstract

The presentation describes a language independent test bench for testing proofing tools, and more generally language technology tools, where the testing is fully automatized. The test results are transformed into xml, and further to HTML. The test bench is freely available as part of the language technology resources in the Divvun project[1] and Centre for Sámi Language Technology at the University of Tromsø[2].

1 Introduction

The development of basic language technology tools for regular end users, such as good spellers and accurate hyphenators, has in practice not progressed since the 1980s, especially within the open source domain. All open source speller engines of today are still list-based as they were in the 70s — they all claim some sort of inheritance from iSpell[3], the (in)famous Unix speller developed originally for English. In the 80-ies the two-level model was developed (Koskenniemi 1983), and further commercialised in proofing tools by Lingsoft[4]. What happened in the 90s was of course the development of grammar checkers based on linguistic analysis, cf. the SCARRIE project (de Smedt & Rosén 2000) and the Constraint Grammar-based grammar checkers from Lingsoft (Birn 2000), but these are not basic tools anymore, and we'll keep them out of the discussion in this article.

There are many reasons for this lack of development, here we will present one cause: the lack of systematic and comparable testing across languages and speller engines to enable easy and automatic comparison of the qualities of available language technologies.

This has led to roughly four tiers in the proofing tools market: 1) good, commercial tools for the big languages – but based on closed source, and with no independent and neutral quality evaluation; 2) reasonably good tools for smaller but rich language societies – still based on closed source and no independent quality assessment; 3) more or less bad tools for many languages, based on open source; and 4) no tools for very many languages.

To help solve this situation, one would need an open, vendor-neutral test bench for proofing tools, together with standardised measures for the quality of these tools. That is what this paper is all about.

2 The Divvun project and automatized testing

The Divvun project develops proofing tools for the Sámi languages, and has so far released spell checkers and hyphenators for North and Julev Sámi. An important secondary goal has been to set up a good, language independent infrastructure to make it easy to add new languages, and an important part of this infrastructure is a good test bench for the tools we make.

In the following we will concentrate on the testing of spell checkers, but we also support testing of hyphenators, and the modular structure of the test bench makes it easy to add support for other tools as well.

The test bench takes three types of input: XML formatted, as tab-separated lines of text, or

[1] http://www.divvun.no/
[2] http://giellatekno.uit.no/
[3] http://fmg-www.cs.ucla.edu/geoff/ispell.html
[4] http://www.lingsoft.fi/

as generated or extracted data from our transducer lexicons. The XML format is used in corpus files for correct-marked documents, and is automatically added from a very simple mark-up system[4] in a copy of the original document. An example of this markup is shown in 1), and the resulting XML is shown in 2).

```
1) Her er ein fiel§(feil).
2) Her er ein <error
correct="feil">fiel<error>.
```

The tab-separated data is used for regression tests, typo tests, and word construction tests. Finally we have a couple of specialised tests to test the conversion from our Xerox-format-based source code to the final proofing tools: baseform tests and paradigm tests.

2.1 Data flow in the test bench

A simple diagram over how the data flows in the test bench is shown in Figure 1.

2.2 How the test results are presented

The test output is read and parsed by a Perl script, and transformed to a standard XML format. From the XML test reports, it is possible to generate all sorts of reports – presently the only supported output is a relatively simple HTML page. In the future we hope to be able to generate overview reports, cross-lingual comparisons, etc.

Although the HTML report is simple, it contains all the relevant statistics for that test run, as well as colour highlighting of essential features. After an introduction with important metadata, the statistics follow, and then the body of the test output from the speller. An example of such a test report can be found on our web site[5].

3 Different speller engines

As seen in Figure 1, we support different speller engines, and it is straightforward to add support for new ones. It is easiest if the speller engine has a command line interface (the test bench is meant to be run from a Unix-like prompt), but it is also possible to script a GUI host application. This is how we run the MS spellers in Word — by using an AppleScript (which can be started from the command line) to script Word, we can run the speller test suit through all languages with built-in speller support in MS Word.

By supporting different spell checkers and spell checker engines, it becomes easy to compare both lexicons and speller engines. We have developed two different versions of our North Sámi Speller, one for MS Office using a speller engine from Polderland[6], and another for OpenOffice, using Hunspell[7] as the speller engine. Table 1 gives Precision and Recall for the two spellers, using a gold standard document as the test data[8].

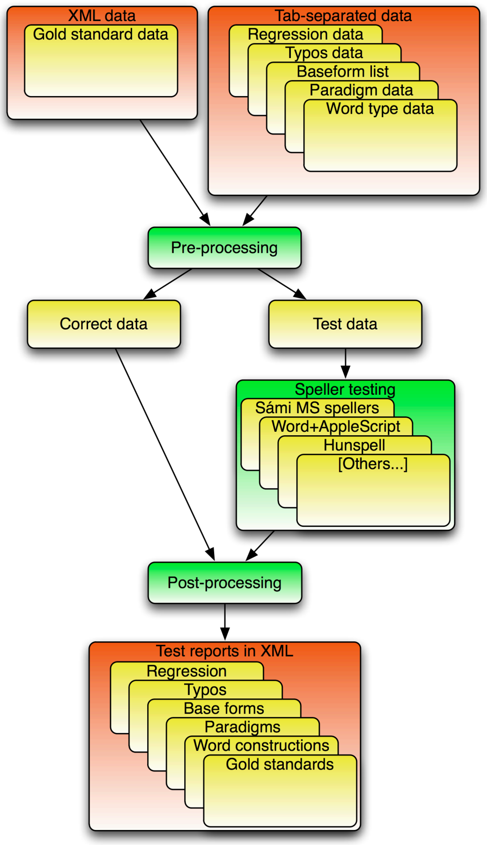


Figure 1: Data flow in the test bench

	Polderland	Hunspell
Precision	89.57	84.07
Recall	98.10	90.48

Table 1: Precision and recall for two North Sámi spell checkers

These figures are of course mainly a measure of how well we have been able to formulate the North Sámi grammar within the limits of the formalism for each spell checker engine. The point in this paper is rather that until now it has been very hard to do such comparisons, while our test bench has turned the task into one simple command on the command line.

[5] http://www.divvun.no/doc/proof/spelling/testing/error-markup.html

[6] http://www.polderland.nl/
[7] http://hunspell.sourceforge.net/
[8] http://www.divvun.no/doc/proof/spelling/testing/Markansluska-pl-forrest-sme-20081013.html

There are other measures of the quality of a spell checker. Table 2 shows the percentage of all spelling errors with: correct suggestion (first row); correct suggestion among the top five suggestions (second row); only incorrect suggestions (third row); and no suggestions (last row). That is, these figures measure the ability to provide relevant suggestions.

	Polderland	Hunspell
Corr sug/all errs	85.44	74.73
Corr sug in top 5	82.52	74.73
Only incorr sug	13.59	25.27
No suggestions	0.97	0.0

Table 2: Suggestion quality for our two North Sámi spellers, gold standard test[7]

In Table 3 the same type of figures is given for another type of test data, a collection of known spelling errors and their corrections[9].

	Polderland	Hunspell
Corr sug/all errs	81.22	72.72
Corr sug in top 5	78.84	72.40
Only incorr sug	15.67	27.28
No suggestions	3.11	0.0

Table 3: Suggestion quality for our two North Sámi spellers when tested on a collection of known typos.

The figures in Table 2 & 3 show that there is a significant difference between the two engines in their ability to provide relevant suggestions. The difference corresponds relatively well to the subjective impression, although I had expected an even bigger difference.

In normal usage the Polderland-based North Sámi speller has a correct suggestion more often than the Hunspell-based one, roughly for 10% more of the spelling errors. Another noticeable difference is that Hunspell never returns nothing – you always get one suggestion or another. In Hunspell's case this means that in regular use as modelled by the gold standard test the speller will suggest just noise in one out of 4 spelling errors. The other speller does the same only in 1 out of 7.

This is a very noticeable difference for the end users. The suggestions are so to speak the user interface of the speller, and the perceived overall quality of the speller will be influenced by the quality of the suggestions. And for minority language writers, the suggestions tend to be more important than for majority language users, since you can expect to find more insecure writers in the minority language community.

[9] http://www.divvun.no/doc/proof/spelling/testing/typos-pl-forrest-sme-20081113.html

4 Further development

The open-source[10] test bench is a work in progress. Among the things we would like to add is support for more speller engines, and other types of proofing tools like grammar checkers. Also, there is much that can be done to extract more statistics and create better reports, as well as to add precision and recall metrics on the suggestions (cf Bick 2006). We would as well like to be able to test more languages.

5 Conclusion

Having access to an open and modular test bench for proofing tools will hopefully be a valuable asset to further develop and improve the most common and important writing aid, the spelling checker. And the possibility to compare different technologies could increase the interest in improving existing tools, and in the best of cases develop new ones. Basic proofing tools are a requirement for supporting small language communities, and the communities deserve better tools than what they are served now. We hope the test bench can be a small contribution in that endeavour.

References

Bick, Eckhard (2006): "A Constraint Grammar Based Spellchecker for Danish with a Special Focus on Dyslexics" in *A Man of Measure – Festschrift in Honour of Fred Karlsson*, pp. 387–396

Birn, Jussi (2000): "Detecting grammar errors with Lingsoft's Swedish grammar checker". In Torbjørn Nordgård (ed.) *NODALIDA '99 Proceedings from the 12th Nordiske datalingvistikkdager*, pp. 28–40. Trondheim: Department of Linguistics, University of Trondheim.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. [PhD dissertation]. Publications of the Department of General Linguistics, University of Helsinki, No. 11.

de Smedt, Koenraad & Victoria Rosén (2000): "Automatic proofreading for Norwegian: The challenges of lexical and grammatical variation". Published in: Nordgård, T. (ed.) *NODALIDA '99: Proceedings from the 12th "Nordiske datalingvistikkdager"*, Trondheim, 9-10 December, 1999 (pp. 206-215). Trondheim: NTNU.

[10] Access to our Subversion repository is protected, but a user name and a password will be given by sending an e-mail to divvun@samediggi.no. Further instructions on our home page, see Footnote 1.

Automatic Variation of Swedish Text by Syntactic Fronting

Kenneth Wilhelmsson
Department of Linguistics
University of Gothenburg
kw@ling.gu.se

Abstract

Ongoing work with a prototype implementation for automatic fronting of primary (main clause) constituents in Swedish input text is described. Linguistic constraints and some technical aspects are also discussed.

1 Introduction

Automatic variation of Swedish text is a relatively unexplored area. Variation by lexical means was tested in an experiment by Rosell (2005) using *Folkets synonymlexikon* (Kann and Rosell, 2005). The program used the fact that synonymy was expressed as a matter of degree (expressed numerically), to vary a threshold value for admitting lexical substitution. The lack of cases of true lexical synonymy, however, seemed to be an important factor, as shown in the evaluation. Producing truth-preserving *(salva veritate)* paraphrases by syntactic means from textual input is a task that has been undertaken in two experimental projects. Pascoe & Ullner (2006) described the process of automatic shift of voice in sentences analyzed by *CassSwe* (Kokkinakis & Johansson Kokkinakis, 1998), producing active sentences from their passive counterparts – a transformation motivated by readability. Lindberg & Svensson (1992) earlier made use of Diderichsen's topological clause description of Nordic languages (Diderichsen, 1946), see table 1. The work dealt with syntactic fronting using a Prolog implementation for achieving truth-preserving variants of hand-picked sentences analyzed by the *MorP Parser* (Källgren, 1992). This paper describes ongoing work with a similar approach to that of the latter, but for free text, using the syntactic analysis described in Wilhelmsson (2008).

Fundamental field	**Nexus field**			**Content field**		
Fundament	**Finite v.**	**Subject**	**Adv.**	**Non-finite v.**	**Obj/ pred.**	**Adv.**
Atomstorleken	*skulle*	*[-]*	*ju*	*peka*		*på motsatsen.*
** Ju*	*skulle*	*atomstorleken*	*[-]*	*peka*		*på motsatsen.*
På motsatsen	*skulle*	*atomstorleken*	*ju*	*peka*		*[-]*
Motsatsen	*skulle*	*atomstorleken*	*ju*	*peka*		*på [-]*

Table 1: An adaptation of Diderichsen's main clause schema showing basic Swedish declarative word order together with fronting of different positional content, including fronting of the prepositional complement *motsatsen* of the adverbial *på motsatsen*.[1]

2 Generation of Paraphrases by Fronting in Input Text

The basic procedure for fronting of any constituent in simple declarative sentences is to place a currently fronted constituent at its *canonical* (or, at least, at an *acceptable*) position according to the sentence schema, whereafter any constituent that it is possible to topicalize may be fronted. The implementation is focused on the task of immediate paraphrase generation in the act of writing to facilitate correct reformulations. It lets a user point at an unbounded full syntactic constituent in the main clause (i.e. subject, object, predicative or adverbial, thus not the fourth example in Table 1), which appear fronted. Thus,

[1] "Prepositional objects" are seen as a type of adverbials, in accordance with e.g. Teleman *et al* (1999).

the parsing is done in parallel with user input. The prototype implementation is made in (uncompiled) JavaScript. The inner representation is an XML-like code, like below.[2]

```
<subjekt>Atomstorleken</subjekt>
<pfv>skulle</pfv>
<adverbial>ju</adverbial>
<adverbial>därmed</adverbial>
<piv>peka</piv>
<adverbial>på motsatsen</adverbial>
<tom>.</tom>
```

A number of restrictions in this straightforward procedure can be noted, of which some are discussed in Lindberg & Svensson (1992).

- Particles, reflexive pronouns and some other primary constituents including a group of adverbials, like *ju* in Table 1 and back-referring expressions (*"vilket var bra"*) cannot be fronted.
- Very long constituents can be fronted, but may make sentences seem clumsy or even unnatural.
- A number of verbs will, if not forming an auxiliary verb construction, as in Table 1, result in a potential violation of the truth-preserving, through subject/object ambiguity. *Bilden föreställer tavlan* will easily introduce a different meaning of a text if transformed into *Tavlan föreställer bilden*.[3]

This type of transformation relies heavily on high accuracy of the syntax analysis, where *exact matching* of primary constituents (including all attributes) is necessary for grammatical output – neither more than one constituent or parts of constituents can be fronted (with a few exceptions such as prepositional complements in Table 1).

A key idea behind the parsing method used is to rely less on matching of unbounded (recursive) primary constituents (subject, object/predicative and adverbials), while identifying bounded ones (e.g., verbs, see Wilhelmsson 2008), thereby delimiting fields in the schema. This particular parsing method, and output format, seems to be appropriate, or even necessary, for the task described. Currently, a POS tagger with an estimated accuracy of 95.3 % is used.

[2] *Pfv* and *piv* here stand for 'primary finite verb' and 'primary non-finite verb', respectively.

[3] Note also, that fronting of a nominal constituent, thereby producing a correct paraphrase, without having made the correct subject/object identification from the start, often is possible.

References

Paul Diderichsen, 1946. *Elementær Dansk Grammatik.* Gyldendal

Viggo Kann and Magnus Rosell, 2005. Free Construction of a Free Swedish Dictionary of Synonyms. In Proc. 15th Nordic Conference on Computational Linguistics – (NODALIDA 05)

Dimitrios Kokkinakis and Sofie Johansson Kokkinakis, 1998, A Cascaded Finite-State Parser for Syntactic Analysis of Swedish, Research Reports from the Department of Swedish, GU-ISS-99-2, University of Gothenburg

Gunnel Källgren, 1992. Making maximal use of surface criteria in large-scale parsing: the MorP parser, Papers from the Institute of Linguistics, University of Stockholm, 60. Stockholm: Univ. Institute of Linguistics.

Janne Lindberg and Carin Svensson, 1992. Topikalisering som skrivstöd. En implementering med satsschema: NADA, KTH.

Maria Pascoe and David Ullner, 2006. VOICEover, Att automatisk aktivera en passiv sats i svenskan, (Datalingvistikprogrammet, University of Gothenburg)

Magnus Rosell, 2005, Automatisk synonymvariering av text (Course project, Språkgranskningsverktyg, KTH)

Ulf Teleman, Erik Andersson and Staffan Hellberg. 1999. *Svenska Akademiens grammatik*

Kenneth Wilhelmsson, 2008. Heuristic Schema Parsing of Swedish Text, The Second Swedish Language Technology Conference (SLTC 2008), KTH, Stockholm