NODALIDA '99

Proceedings from the 12th "Nordiske datalingvistikkdager".

Trondheim, 9-10 December, 1999.

Torbjørn Nordgård, editor

Trondheim

Department of Linguistics, NTNU

2000

This volume and the conference were sponsored by:
Telenor
Faculty of Arts, NTNU
Department of Linguistics, NTNU

Published by
Department of Linguistics
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

Printed by
NTNU

ISBN 82-92225-00-5

Contents

BusTUC - A natural language bus route adviser in Prolog

Tore Amble
Knowledge Systems Group
Department of Computer and Information Science
NTNU
amble@idi.ntnu.no

25. april 2000

Sammendrag

The paper describes a natural language based expert system route adviser for the public bus transport in Trondheim, Norway. The system is available on the Internet, and has been installed at the bus company's web server since the beginning of 1999. The system is bilingual, relying on an internal language independent logic representation.

1 Introduction

A natural language interface to a computer database provides users with the capability of obtaining information stored in the database by querying the system in a natural language (NL). With natural language as a means of communication with a computer system, the users can make a question or a statement in the way they normally think about the information being discussed, freeing them from having to know how the computer stores or processes the information.

The present implementation represents a a major effort in bringing natural language processing into practical use. A system is developed that can answer queries about bus routes, stated in natural language texts, and made public through the Internet World Wide Web (`www.idi.ntnu.no/bustuc/`).

Trondheim is a small city with a university and 140000 inhabitants. The central bus system in Trondheim has 42 bus lines, serving 590 stations, with

1900 departures per day (in average). That gives approximately 60000 scheduled bus station passings per day, which is somehow represented in the route data base.

The starting point is to automate the function of a route information agent. The following example of a system response is taken from an actual request over telephone to the local route information company:

```
Hi, I live in Nidarvoll and tonight I must
reach a train to Oslo at 6 oclock.
```

A typical answer would follow quickly:

```
-----------------------------------------------
Bus number 54 passes by Nidarvoll school at 1710
and arrives at Trondheim Railway Station at 1725.
-----------------------------------------------
```

In between the question and the answer is a process of lexical analysis, syntax analysis, semantic analysis, pragmatic reasoning and database query processing and answer generation.

One could argue that the information content could be solved by an interrogation, whereby the customer is asked to produce 4 items: `departure station`, `arrival station`, `earliest` and `latest arrival time`. It is a myth that natural language is better way of communication because it is "natural language". The challenge is to prove by demonstration that an NL system can be made that will be preferred to the interrogative mode. To do that, the system has to be correct, user friendly and almost complete within the actual domain.

2 Previous Efforts, CHAT-80, PRAT-89 and HSQL

The system, called BusTUC is built upon the classical system CHAT-80 ([WP82]). CHAT-80 was a state of the art natural language system that was impressive on its own merits, but also established Prolog as a viable and competitive language for Artificial Intelligence in general. The system was a brilliant masterpiece of software, efficient and sophisticated. The natural language system was connected to a small query system for international geography. The following query could be analysed and answered in less than half a second:

```
Which country bordering the Mediterranean borders a
country that is bordered by a country whose population
exceeds the population of India?
```

(*The answer 'Turkey' has become incorrect as time has passed. The irony is that Geography was chosen as a domain without time.*)

The ability to answer ridiculously long queries is of course not the main goal. The main lesson is that complex sentences are analysed with a proper understanding without sacrificing efficiency. Any superficial pattern matching technique would prove futile sooner or later.

2.1 Making a Norwegian CHAT-80, PRAT-89

At the University of Trondheim (NTNU), two students made a Norwegian version of CHAT-80,called PRAT-89 ([TV88],[TV89]). (Also, a similar Swedish project SNACK-85 was reported).

The dictionary was changed from English to Norwegian together with new rules for morphological analysis. The change of grammar from English to Norwegian proved to be amazingly easy. It showed that the langauges were more similar than one would believe, given that the languages are incomprehensible to each other's communities.

After changing the dictionary and grammar, the following Norwegian query about the same domain could be answered correctly in a few seconds.

```
Hvilke afrikanske land som har en befolkning større
enn 3 millioner og mindre enn 50 millioner og er nord
for Botswana og øst for Libya har en hovedstad som
har en befolkning større enn 100 tusen ?
```

(*"Which African countries that have a population greater than 3 millions and less than 50 millions and is north of Botswana and east of Libya has a capital which has a population greater than 100 thousands ?")*

2.2 HSQL - Help System for SQL

A Nordic project HSQL (Help System for SQL) was accomplished in 1988-89 to make a joint Nordic effort interfaces to databases.

The HSQL project was led by the Swedish State Bureau (Statskontoret), with participants from Sweden, Denmark, Finland and Norway [AKL+90]. The aim of HSQL was to build a natural language interface to SQL databases for the Scandinavian languages Swedish, Danish and Norwegian. These

languages are very similar, and the Norwegian version of CHAT-80 was easily extended to the other Scandinavian languages. Instead of Geography, a more typical application area was chosen to be a query system for hospital administration. We decided to target an SQL database of a hospital administration which had been developed already.

The next step was then to change the domain of discourse from Geography to hospital administration, using the same knowledge representation techniques used in CHAT-80. A semantic model of this domain was made, and then implemented in the CHAT-80 framework.

The modelling technique that proved adequate was to use an extended Entity Relationship (ER) model with a class (type) hierarchy, attributes belonging to each class, single inheritance of attributes and relationships.

Connecting the system to an SQL database.

After the remodelling, the system could answer queries in "Scandinavian" to an internal hospital database as well as CHAT-80 could answer Geography questions. HSQL produced a Prolog-like code FOL (First Order Logic) for execution. A mapping from FOL to the data base Schema was defined, and a translator from FOL to SQL was implemented. The example

```
Hvilke menn ligger i en kvinnes seng?
```

(*"Which men lie in a woman's bed?"*)

was translated dryly into the SQL query:

```
SELECT DISTINCT T3.name,T1.sex,T2.reg_no,T3.sex,
       T4.reg_no,T4.bed_no,T5.hosp_no,T5.ward_no
FROM PATIENT T1,OCCUPANCY T2,PATIENT T3,
     OCCUPANCY T4,WARD T5
WHERE (T1.sex='f') AND (T2.reg_no=T1.reg_no) AND
      (T3.sex='m') AND (T4.reg_no=T3.reg_no) AND
      (T4.bed_no=T2.bed_no) AND (T5.hosp_no=T4.hosp_no) AND
      (T5.ward_no=T4.ward_no)
```

2.3 The Understanding Computer

The HSQL was a valuable experience in the effort to make transportable natural language interfaces. However, the underlying system CHAT-80 restricted the further development.

After the HSQL Project was finished, an internal reseach project TUC (The Understanding Computer) was initiated at NTNU to carry on the results from HSQL. The project goals differed from those of HSQL in a number of ways, and would not be concerned with multimedia interfaces. On the other hand, portability and versatility were made central issues concerning the generality of the language and its applications. The research goals could be summarised as to

- Give computers an operational understanding of natural language.
- Build intelligent systems with natural language capabilities.
- Study common sense reasoning in natural language.

A test criterion for the understanding capacity is that after a set of definitions in a Naturally Readable Logic, NRL, the system's answer to queries in NRL should conform to the answers of an idealised rational agent.

```
Every man that lives loves Mary. John is a man. John lives.
Who loves Mary?
==> John
```

NRL is defined in a closed context. Thus interfaces to other systems are in principle defined through simulating the environment as a dialogue partner.

TUC is a prototypical natural language processor for English written in Prolog. It is designed to be a general purpose easily adaptable natural language processor. It consists of a general grammar for a subset of English, a semantic knowledge base, and modules for interfaces to other interfaces like UNIX, SQL-databases and route information services.

2.4 The TABOR Project

It so happened that a Universtity project was starteded in 1996, called TABOR ("Speech based user interfaces and reasoning systems"), with the aim of building an automatic public transport route oracle, available over the public telephone. At the onset of the project, the World Wide Web was fresh, and not as widespread as today, and the telephone was still regarded as the main source of information for the public. Since then, the Internet has become the dominant medium, and it is as likeley to find a computer with Internet connection, as finding a telephone, or a local busroute booklet for that matter.

It was decided that a text based information system should be built, regardless of the status of the speech rocgnition and speech synthesis effort, which proved to lag behind after a while.

The BusTUC system

The resulting system BusTUC grew out as a natural application of TUC, and an English prototype could be built within a few months ([Bra97]). Since the summer 1996, the prototype was put onto the Internet, and has been developed and tested more or less continually since then. The most important extension was that the system was made bilingual (Norwegian and English) during the fall 1996.

In the spring 1999, the BusTUC was finally adopted by the local bus company in Trondheim (A/S Trondheim Trafikkselskap), which set up a server (300 MHz PC with Linux).

Until today, over 150.000 questions have been answered, and BusTUC seems to stabilize and grow increasingly popular.

3 Anatomy of the bus route oracle

The main components of the bus route information systems are:

- A parser system, consisting of a dictionary, a lexical processor, a grammar and a parser.
- A knowledge base (KB), divided into a semantic KB and an application KB
- A query processor, containg a routing logic system, and a route data base.

The system is bilingual and contains a double set of dictionary, morphology and grammar. Actually, it detects which language is most probable by counting the number of unknown words related to each language, and acts accordingly. The grammars are surprisingly similar, but no effort is made to coalesce them. The Norwegian grammar is slightly bigger than the English grammar, mostly because it is more elaborated but also because Norwegian allows a freer word order.

3.1 Features of BussTUC

For the Norwegian system, the figures give an indication of the size of the domain: 420 nouns, 150 verbs, 165 adjectives, 60 prepositions, etc.

There are 1300 grammar rules (810 for English) although half of the rules are at a low lexical level.

The semantic net described below contains about 4000 entries.

A big name table of 3050 names in addition to the official station names, is required to capture the variety of nameings. A simple spell correction is part of the system (essentially 1 character errors).

The pragmatic reasoning is needed to translate the output from the parser to a route database query language. This is done by a production system called Pragma, which acts like an advanced rewriting system with 580 rules.

In addition, there is another rule base for actually generating the natural language answers (120 rules).

The system is mainly written in Prolog (Sicstus Prolog 3.7), with some Perl programs for the communication and CGI-scripts.

At the moment, there are about 35000 lines of programmed Prolog code (in addition to route tables which are also in Prolog). Sicstus Prolog proved to be extremely efficient and reliable for the application.

Average response time is usually less than 2 seconds, but there are queries that demand up to 10 seconds.

The error rate for the single, correct, complete and relevant questions is about 2 percent.

3.2 The Parser System

The Grammar System

The grammar is based on a simple grammar for statements, while questions and commands are derived by the use of movements. The grammar formalism which is called Consensical Grammar, (CONtext SENSItive CompositionAL Grammar) is an easy to use variant of Extraposition Grammar ([PW80]), which is a generalisation of Definite Clause Grammars. Semantically, a phrase is composed of the semantics of the subphrases; the basic constituents being generalized verb complements. As for Extraposition grammars, a grammar is translated to Definite Clause Grammars, and executed as such.

A characteristic syntactic expression in Consensical Grammar may define an incomplete construct in terms of a "difference " between complete constructs. This implements various kinds of movements by using the subtracted parts instead of reading from the input, immediately or after a gap.

The effect is the same as for Extraposition grammars, but this format allows a more intuitive reading. Examples of grammar rules:

```
statement(P) --->
   noun_phrase(X,VP,P),
   verb_phrase(X,VP).

statement(Q) --->
```

```
    verb_complements0(VC),  % initial optional verb_complements
    statement(Q) -...       % may be  inserted  after a gap
         verb_complements0(VC).

whoseq(P) --->   % whose dog barked?
    [whose],
    noun(N),
    whoq(P) - ([who],[has],[a],noun(N),[that]). % without gap

whoq(P) --->
    [who],
    whichq(P) - ([which],[person]).

whichq(which(X)::P) --->
    [which],
    statement(P) - the(X).
```

Example:

```
  Whose dog barked?
```

is analysed as if the sentence had been

```
Who has a dog that barked?
```

which is analysed as

```
Which person has a dog that barked?
```

which is analysed as

```
for which X is it true that
the (X) person has a dog that barked?
```

where the last line is analysed as a `statement`.

Movement is easily handled in Consensical Grammar without making special phrase rules for each kind of movement. The following example shows how TUC manages a variety of analyses using movements:

```
Max said Bill thought Joe believed Fido Barked.

Who said Bill thought Joe believed Fido barked?    ==> Max
Who did Max say thought Joe believed Fido barked?  ==> Bill
Who did Max say Bill thought believed Fido barked? ==> Joe
```

The parser

The experiences with Consensical grammars are a bit mixed however. The main problem is the parsing method itself, which is top down with backtracking. Many principles that would prove elegant for small domains turned out to be to costly for larger domains, due to the wide variety of modes of expressions, the incredible ambiguity and the sheer size of the covered language.

These problems also made it imperative to introduce a timeout on the parsing process of embarassing 10 seconds. Although most sentences would be parsed within a second, some legal sentences of moderate size actually need this time.

The disambiguation is a major problem for small grammars and large languages, and was solved by the following guidelines:

- a semantic type checking was integrated into the parser, and would help to discard semantically wrong parses from the start.
- a heuristics proved almost irreproachable: The longest possible phrase of a category that is semantically correct is in most cases the preferred interpretation.
- due to the perplexity of the language, some committed choices (cuts) had to be inserted into the grammar at strategic places. As one could fear however, this implied that wrong choices being made at some point in the parsing could not be recovered by backtracking.

3.3 The semantic knowledge base

Adaptability means that the system does not need to be reprogrammed for each new application.

The design principle of TUC is that most of the changes are made in a tabular semantic knowledge base, while there is one general grammar and dictionary. In general, the logic is generated automatically from the semantic knowledge base.

The nouns play a key role in the understanding part as they constitute the class or type hierarchy. Nouns are defined in an `a-kind-of` hierarchy. The hierarchy is tree-structured with single inheritance. The top level also constitute the top level ontology of TUC's world.

In fact, a type check of the compliances of verbs, nouns adjectives and prepositions is not only necessary for the semantic processing but is essential for the disambiguation in the syntax analysis. In TUC, a declaration

of the legal combinations are carefully assembled in the semantic network, which then serves a dual purpose. These semantic definitions are necessary for disambiguating prepositional attachments, for instance in the following sentences

```
The dog saw a man with a telescope.
The man saw a dog with a telescope.
```

to be treated differently because `with telescope` may modify the noun `man` but not the noun `dog`, while `with telescope` modifies the verb `see`, restricted to person.

3.4 The Query Processor

Event Calculus

The semantics of the phrases are built up by a kind of verb compelements, where the event play a central role.

The text is translated from Natural language into a form called TQL (Temporal Query Language/TUC Query Language) which is a first order event calculus expression, a self contained expression containing the literal meaning of an utterance.

The formalism TQL that was defined, inspired by the Event Calculus by Kowalski and Sergot ([KS86]). The TQL expressions consist of predicates, functions, constants and variables. The textual words of nouns and verbs are translated to generic predicates using the selected interpretation. The following question

```
Do you know whether the bus goes to Nidarvoll on Saturday ?
```

would give the TQL expression below. Typically, the Norwegian equivalent

```
Vet du om bussen går til Nidarvoll på søndag ?
```

gives exactly the same code.

```
test::                          % Type of question
  isa(real,program,bustuc),     % bustuc is a real program
  isa(real,bus,A),              % A is a real bus
  isa(real,saturday,B),         % B isa saturday
  isa(real,place,nidarvoll),    % nidarvoll isa place
  event(real,D),                % D is an event
  know(id,whether,bustuc,C,D),  % C is a statement known at D
  event(C,E),                   % E is an event in C
```

```
action(go,E),                  % the action of E is 'go'
actor(A,E),                    % the actor of E is  A
srel(to,place,nidarvoll,E),    % E is related to nidarvoll
srel(on,time,B,E).             % E is related on the saturday B
```

The event parameter plays an important role in the semantics. It is used for various purposes. The most salient role is to identify a subset of time and space in which an action or event occured. Both the actual time and space coordinates are connected to the actions through the event parameter.

Pragmatic reasoning

The TQL is translated to a route database query language (BusLOG) which is actually a Prolog program. This is done by a production system called Pragma, which acts like an advanced rewriting system with 580 rules.

4 Conclusions

The TUC approach has as its goal to automate the creation of new natural language interfaces for a well defined subset of the language and with a minimum of explicit programming.

The implemented system has proved its worth, and is interesting if for no other reason. There is also an increasing interest from other bus companies and route information companies alike to get a similar system for their customers.

Further work remains to make the parser really efficient, and much work remains to make the language coverage complete within reasonable limits.

It is an open question whether the system of this kind will be a preferred way of offering information to the public.

If it is, it is a fair amount of work to make it a portable system that can be implemented elsewhere, also connecting various travelling agencies.

If not, it will remain a curiosity. But anyway, a system like this will be a contribution to the development of intelligent systems.

Referanser

[AKL+90] Tore Amble, Erik Knudsen, Aarno Lehtola, Jan Ljungberg, and Ole Ravnholt. *Naturlig Språk och Grafik - nya vägar inn i databaser.* Statskontoret, 1990. Rapport om HSQL, ett kunskapsbaseret hjälpsystem för SQL.

[Bra97] Jon S. Bratseth. BusTUC - A Natural Language Bus Traffic Informations System. Master's thesis, The Norwegian University of Science and Technology, 1997.

[KS86] R. Kowalski and M. Sergot. A logic based calculus of events. *New Generation Computing*, 8(0):67–95, 1986.

[PW80] F.C.N. Pereira and D.H.D. Warren. Definite clause grammar for language analysis. *Artificial Intelligence*, 0(3), 1980.

[TV88] J. Teigen and V. Vetland. Syntax analysis of norwegian language. Technical report, The Norwegian Institute of Technology, 1988.

[TV89] J. Teigen and V. Vetland. Handling reasonable questions beyond the linguistic and conceptual coverage of
natural language interfaces. Master's thesis, The Norwegian Institute of Technology, 1989.

[WP82] D.H.D Warren and F.C.N. Pereira. An efficient and easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4), 1982.

DEVELOPING A GRAMMAR CHECKER FOR SWEDISH[1]

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

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

1. Introduction

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

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

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

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

2. What is a grammar checker – really?

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

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

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

There is somewhat of a confusion or at least vagueness in the general consciousness of what grammar checkers are as proofing tools. Grammar checkers are often not, despite their name, only limited to purely grammatical or, to be specific again, syntactic features. In addition to these errors, grammar checkers typically address violations of or non-conformances with established conventions in punctuation, word capitalization, and number and date formatting. Furthermore, word-specific stylistic assessments are often

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

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

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

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

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

error detection rules (cf. Birn in this volume). Furthermore, the goal was to develop the end-product within a preset timeframe, which required the prioritization in the implementation of possible error types.

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

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

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

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

The development of Grammatifix was originally started out as an exploratory project. At the very beginning, existing grammar checkers for other languages were investigated, both for the linguistic features that they covered and how well they performed their tasks, an activity that seems to have been undertaken by other projects (e.g. SCARRIE)[6]. After this, a general classification of linguistic error types, writing style violations and non-recommended word usage that were judged worth finding was

compiled, using the linguistic intuition or personal observations of project members[7] and generally acknowledged guide and reference books of Swedish grammar and writing conventions. All reference works consulted at this phase were of Sweden-Swedish (i.e. *"riksvensk"*) origin. From the very beginning, Swedish material that the company or individual project members had access to, ranging from personal observations of errors in newspapers to actual corpora of Swedish texts at the company's disposal, was used to support this classification work by providing a source of genuine evidence for the existence and character of hypothesized error types, and for the discovery of new ones. These genuine examples would grow to form the kernel of the error corpus later used in the development and testing of the linguistic error detection rules (cf. Birn in this volume). After this stage, each error type in this classification was evaluated along two criteria. Firstly, the existence of a Lingsoft-proprietary technology (e.g. SWECG) or a public one (such as regular expression matching techniques), or any known technology or technique for that matter that could be used to detect the particular error type was assessed. Secondly, the perceived benefit and consequent priority of detecting a particular error type was evaluated.

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

The list of the error types addressed by Grammatifix should consequently be of no great surprise, and is rather similar to those of the other projects, which can naturally be attributed to the language in question. Thus, checks on noun phrase internal agreement and verb chain consistency have a central place in the error type portfolio. All in all, Grammatifix covers 43 error type checks, of which 26 are syntactic in nature (of which 17 belong either to the noun phrase internal or to verb chain consistency error types), 14 address punctuation, number and date formatting conventions, and 3 cover word-

specific non-standard stylistic usage. A more specific listing of these error types with example errors is given in Table 1 (syntactic errors) and Table 2 (non-syntactic errors). Since Grammatifix is under constant development, an up-to-date version of its error types is available on the Internet (Arppe et al 1999).

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

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

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

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

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

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

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

5. Should something happen after error detection?

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

communicates its linguistic findings to the end-user that the quality of the grammar checker is ultimately perceived by him or her. In the section below, emphasis is put on the treatment of syntactic errors, though the same principles have been applied to the other error types, too.

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

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

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

Finally, a suggestion for the correction of the suspected erroneous word is provided, when practically feasible. In most error types, the suggestion is generated by applying the error correction scheme, representing the spirit of the error diagnosis text, to the

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

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

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

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

In addition to the error detection and error treatment modules, a set of longer explanation texts are incorporated into Grammatifix, covering concisely the central syntactic and non-syntactic issues related with the targeted error types. Furthermore,

each time after Grammatifix has gone through a selected text, it provides some general statistics, including the number of characters, words, sentences and paragraphs, the average number of characters per words, the average number of words per sentence, and the average number of sentences per paragraph in the checked text. In conjunction with these statistics, Grammatifix also provides a so-called LIX value (*"Läsbarhetsindex"*), which is a readability index developed for Swedish.[9] It should be noted that when used as an integrated module it is up to the client application which components, that have been described in this section and that are provided by Grammatifix, are indeed conveyed to the end-user.

6. Comparison with the other known grammar checkers for Swedish

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

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

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

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

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

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

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

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

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

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

Notes

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

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

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

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

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

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

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

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

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

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

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

References:

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

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

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

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

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

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

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

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

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

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

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

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

DETECTING GRAMMAR ERRORS WITH LINGSOFT'S SWEDISH GRAMMAR CHECKER

Juhani Birn
Lingsoft, Inc.
jbirn@lingsoft.fi

Abstract

A Swedish grammar checker (Grammatifix) has been developed at Lingsoft. In Grammatifix, the Swedish Constraint Grammar (SWECG) framework has been applied to the task of detecting grammar errors. After some introductory notes (chapter 1), this paper explains how the SWECG framework has been put to use in Grammatifix (chapter 2). The different components of the system (section 2.1) and the formalism of the error detection rules (section 2.2) will be overviewed, and the relationship between grammar errors and disambiguation will be discussed (section 2.3). Work on the avoidance of false alarms is also described (chapter 3). Finally, test results are reported (chapter 4).

1. Introduction

The purpose of this paper is to explain how Grammatifix goes about its task of detecting grammar errors. The paper by Arppe (this volume) addresses the more general level design principles in the development of Grammatifix, and provides also a background to the field of Swedish grammar checking in general.

Grammatifix has checks on three kinds of phenomena: grammar errors, graphical writing convention errors, and stylistically marked words.[1] For these phenomena different detection techniques are used: SWECG, matching of regular expressions against character sequences, and lexical tagging, respectively. This paper is concerned with grammar error detection.

Prototypical grammar errors can be understood to be norm violations that are to be identified in contexts larger than the word (cf. spell-checking) where the contexts are morphosyntactically explainable. Of errors so defined, no computational grammar checker is able to control more than a (more or less) modest part. A realistic grammar checker concentrates on central categories of the language's grammar, and, within those categories, on common, simple patterns that allow precise descriptions. The error categories targeted by Grammatifix are presented in Arppe & al. (1999), for a listing with examples see also Arppe (this volume).

2. Constraint Grammar as a framework for grammar error detection

Constraint Grammar (CG) is a framework for part-of-speech disambiguation and shallow syntactic analysis, as originally proposed by Karlsson (1990). The basic principles and the formalism of CG are fully explained in Karlsson & al. (1995). A short presentation of SWECG is given in Birn (1998). In Grammatifix, the CG framework is used for the purposes of grammar error detection.

2.1. Overview of the error detector's components

The CG-based error detection system consists of five sequential components as listed below (1-5). In a formal sense the componets are the same as in SWECG, but, contentwise, the components of the two systems are not identical. There are some differences even in components (1, 2), some more in component (3), and components (4, 5) are wholly application-specific.

(1) Preprocessing
(2) Lexical analysis
(3) Disambiguation
(4) Assignment of the tags @ERR and @OK to each word
(5) Error detection rules, i.e. rules for the selection of @ERR

Preprocessing. The preprocessor (or tokeniser) identifies words, abbreviations, punctuation marks, and fixed syntagms. A fixed syntagm is a multi-word expression identified as a lexical unit, e.g. the words *till hands* are identified as a unit, *till_hands*, analysed as an ADV[2]. This treatment entails that the error detector avoids false alarms that might follow (in unexpected contexts, e.g. *funnits till hands dygnet om*) if a genitive feature was present in the analysis of *till hands*.

The tasks performed by componets (2–5) will be illustrated with a stepwise analysis of the relevant (here boldfaced) parts of the example sentence given below. The error to be detected is the definite form *stavningen* as governed by the genitive *vilkas*. The analysis of the sequence *många engelska* also illustrates a relevant point.

Det finns ***många engelska lånord vilkas diskontinuerliga stavningen****inte tycks bereda språkbrukarna några problem.*
(From *Språket lever. Festskrift till Margareta Westman*. Norstedts 1996:68.)

Lexical analysis. The main module here is the SWETWOL analyser (Karlsson 1992; cf. also Birn 1998). As illustrated below each word is here given one or more readings. For example, *många* has two readings, DET (implying modifier status) and PRON (implying head word status), and *engelska* has three readings, one of them N SG. The sequence *många engelska* illustrates why it was obvious from the start that disambiguation should be used: *många* is **PL** and *engelska* is N **SG** (inter alia), but flagging this as a number agreement error would be a false alarm, of course. Disambiguation is needed for the sake of precision.

```
"<många>"
    "mången" <ID> DET UTR/NEU INDEF PL NOM
    "mången" PRON UTR/NEU INDEF PL NOM
"<engelska>"
    "engelsk" A UTR/NEU DEF SG NOM
    "engelsk" A UTR/NEU DEF/INDEF PL NOM
    "engelska" N UTR INDEF SG NOM
"<lånord>"
    "lån_ord" N NEU INDEF SG/PL NOM
"<vilkas>"
    "vilken" <WH> <CLB> <MD> DET UTR/NEU INDEF PL GEN
    "vilken" <WH> <CLB> PRON UTR/NEU INDEF PL GEN
"<diskontinuerliga>"
    "diskontinuerlig" A UTR/NEU DEF SG NOM
    "diskontinuerlig" A UTR/NEU DEF/INDEF PL NOM
"<stavningen>"
    "stavning" N UTR DEF SG NOM
```

Disambiguation. The disambiguation rules of SWECG have been adopted to a large extent as such in Grammatifix, but, importantly, there are differences. The differences are a consequence of the efforts, in Grammatifix, to overcome certain disambiguation disturbances due to grammar errors (for more on this point see section 2.3). Full disambiguation is not a goal as such for Grammatifix, and some of the error detection rules are formulated so as to tolerate ambiguities or even incorrect disambiguations (section 2.3). In the example sentence of this section, the disambiguator selects the appropriate reading for each word, e.g. *engelska* is disambiguated as A PL as shown below.

```
"<många>"
    "mången" <ID> DET UTR/NEU INDEF PL NOM
"<engelska>"
    "engelsk" A UTR/NEU DEF/INDEF PL NOM
"<lånord>"
    "lån_ord" N NEU INDEF SG/PL NOM
```

Assignment of the tags @ERR and @OK to each word. In ordinary CG the component called 'Morphosyntactic mappings' assigns a number of syntactic tags (subject, object, premodifier, etc.)

to each remaining reading. In Grammatifix this component performs a trivial task: each reading is assigned two more tags, @ERR (error) and @OK (no error), as shown below for *många*.

```
"<många>"
    "mången" <ID> DET UTR/NEU INDEF PL NOM @ERR @OK
```

Error detection rules, i.e. rules for the selection of @ERR. In ordinary CG the component called 'Syntactic constraints' performs syntactic disambiguation, i.e. there are rules that try to select the contextually appropriate syntactic tags. In Grammatifix this component contains error detection rules, i.e. rules for the selection of the tag @ERR for those words where an error can be located. In the example, @ERR lands on *stavningen*, and all other words get @OK. The words with @ERR, possibly together with some of the surrounding words, are flagged to the user.

```
"<många>"
    "mången" <ID> DET UTR/NEU INDEF PL NOM  @OK
"<engelska>"
    "engelsk" A UTR/NEU DEF/INDEF PL NOM @OK
"<lånord>"
    "lån_ord" N NEU INDEF SG/PL NOM @OK
"<vilkas>"
    "vilken" <WH> <CLB> <MD> DET UTR/NEU INDEF PL GEN  @OK
"<diskontinuerliga>"
    "diskontinuerlig" A UTR/NEU DEF SG NOM @OK
"<stavningen>"
    "stavning" N UTR DEF SG NOM @ERR
```

The selection of @ERR is performed by rules which use the CG disambiguation rule formalism (section 2.2). For the above case the rule is in basic outline as shown below. This formulation, a formally valid CG rule, is simplified in the sense that here are not included any of the additional conditions used for the avoidance of false alarms (chapter 3).

```
Error detection rule (simplified):
(@w =s! (@ERR)          ;Read: For a word (@w), select (=s!) the error tag (@ERR),
        (0 N-DEF)       ;if the word itself is a noun in definite form (0 N-DEF), and
        (-2 GEN)        ;if the second word to the left is a genitive (-2 GEN), and
        (-1 A-DEF))     ;if the first word to the left is an adjective in definite form (-1 A-DEF).
```

The current description contains 659 @ERR rules. After all the @ERR rules have been tried, there is one final "rule" that picks @OK for all the remaining words. (No word has the feature DUMMY referred to in the rule.)

```
(@w =s! (@OK)               ;Read: For a word (@w), select (=s!) the @OK tag,
        (NOT 0 DUMMY))      ;if the word does not have the feature DUMMY.
```

What the actual CG components are used for in Grammatifix has been explained above. – To each @ERR rule is attached (a number that refers to) an error message. An error message consists of an error title, a short explanation, a correction scheme (when possible), and (behind a button) a longer explanation of the grammar point mentioned in the title. Below is given the error message, except for the longer explanation, attached to the @ERR rule presented above. Triggered by the above example sentence, the position slots (0) and (-2) in the explanation are filled by the words *stavningen* and *vilkas*, respectively. The correction means that the DEF form of the noun in position (0) is transformed into INDEF, so the correction suggested to the user is *stavning*.

Error title: *Substantivets bestämdhetsform*
Explanation: *Kontrollera ordformen (0). Om ett substantiv styrs av en genitiv, t.ex. (-2), bör det stå i obestämd form.*
Correction: *(0 N DEF) => (0 N INDEF)*

2.2. Overview of the error detection rule formalism

As noted, Grammatifix error detection (i.e. @ERR selection) rules use the CG rule formalism. For a full explication of the CG rule formalism see chapter 2 in Karlsson & al. (1995) – as a companion to the study of that chapter 2, below is given a convenient overview of the rule formalism as applied to @ERR selection. The example rule is already familiar (see section 2.1). After the overview follow some more examples of the ways in which the formalism can be used for error detection.

A Constraint Grammar error detection rule consists of four parts:

	Domain	**Operator**	**Target**	**Context condition(s)**
Example:	(@w	=s!	(@ERR)	(0 N-DEF) (-2 GEN) (-1 A-DEF))

Where:

Domain: @w (any word-form) or "<...>" (a specific word-form, e.g. "<ett>").
Operator: =s! (select) or =s0 (remove)
Target: @ERR or @OK.
Context condition: Polarity Position(Careful-mode) Set (Linked-position).

Polarity: Positive or negative (NOT). Examples:
(1 N) = the word in position 1 is N (i.e. has a N reading).
(NOT 1 N) = the word in position 1 is not N (i.e. does not have a N reading).

Position:

Target: 0.
Absolute: 1, 2, 3 etc., and -1, -2, -3 etc., in relation to the target. Examples:
(1 V) = the first word to the right from the target is V.
(-2 V) = the second word to the left from the target is V.
Unbounded: *1, *2, *3 etc., and *-1, *-2, *-3 etc., in relation to the target. Examples:
(*1 V) = a V one or more words rightwards from the target.
(*-2 V) = a V two or more words leftwards from the target.
Linked: R+1, R+2, R+3 etc. and *R, and L-1, L-2, L-3 etc. and *L, starting from a word found in some unbounded '*' position. Examples:
(*1 V R+1) (R+1 N) = somewhere to the right (*1) from the target is found a V, and the next word to the right (R+1) from that V is an N (R+1 N).
(*1 V L-1) (L-1 N L-1) (L-1 A) = somewhere to the right (*1) from the target is found a V, and the next word to the left (L-1) from that V is an N (L-1 N), and the next word to the left (L-1 again) from that N is an A (L-1 A). (Several linkings are possible.)
(*-1 AUX *R) (NOT *R INF) = somewhere to the left (*-1) from the target is found an AUX and to the right (*R) from that AUX there is no INF preceding the target (NOT *R INF).

Careful mode: A position may have C for 'careful mode', meaning that the condition is satisfied only in an unambiguous context. Example:
(1C N) = the word in position 1 has no other readings than N.

Set: Anything referred to in the context conditions must initially be declared as a set. Examples:

Set	**Set elements**
(GEN	GEN)
(N-NEU	(N NEU))
(A-DEF	(A DEF) (A DEF/INDEF))
(MOD-AUX	"kunna" ("vilja" V) ...)

Below are given four more illustrations of the error detection properties of the rule formalism. The rules here are simplified in the same sense as the @ERR rule in section 2.1, i.e. we ignore here the additional (sometimes highly specific) context conditions used for false alarm avoidance in the real rules. – The first rule below illustrates that the **domain** of a rule can be a specific **word form**, in this case "<ett>". The C as in 1C stands for **careful mode** (unambiguous analysis required), used in a majority of the @ERR rule context conditions.

Example: *Ett*(@ERR) *högtrycksrygg förskjuts norrut.*

Error detection rule (simplified):

```
("<ett>" =s! (@ERR)       ;Read: For the word-form Ett/ett, select (=s!) the error tag (@ERR),
        (1C N-UTR))  ;if the next word to the right is an unambiguous utrum noun (1C N-UTR).
```

The above rule uses a (maximally) close context. The following rules show that you can also refer to more comprehensive contexts. You may want to check e.g. that the whole sentence lacks some feature, e.g. the feature 'finite verb' as in the rule below. The rule illustrates **unbounded positions**, in this case *-1 (anywhere leftwards starting from position -1) and *1 (anywhere rightwards starting from position 1). **Negative** conditions (NOT) are often crucial.

Example: *Pulsen bli*(@ERR) *för kraftig.*

```
Error detection rule (simplified):
(@w =s! (@ERR)              ;Read: For a word (@w), select (=s!) the error tag (@ERR),
        (0C V-INF)          ;if the word itself is anunambiguous infinitive (0C V-INF), and
        (NOT *-1 V-FIN)     ;if there is no finite verb to the left (NOT *-1 V-FIN), and
        (NOT *1 V-FIN))     ;if there is no finite verb to the right (NOT *1 V-FIN).
```

A common situation is that you want to restrict a search to some specified portion of the sentence. This is illustrated by the last two examples. In the next example, the rule checks that there is no verb (especially, no instance of *ha*) between *skulle* and *skrivits*.[3] The rule illustrates the use of **linked conditions**, the link provided by an identical hook in the conditions, in this case by *R. Starting from a supine (*skrivits*) the first search here is for a modal auxiliary to the left and, when the first instance (*skulle*) is found, the second search starts for the non-occurrence of verbs between the modal auxiliary and the supine.

Example: *Så kom en flicka, som Göran höll av, in på Bibliotekshögskolan i Borås och hade hon inte gjort det skulle kanske denna artikel aldrig skrivits*(@ERR).

```
Error detection rule (simplified):
(@w =s! (@ERR)              ;Read: For a word (@w), select (=s!) the error tag (@ERR),
        (0C V-SUPINE)       ;if the word itself is un unambiguous supine (0C V-SUPINE), and
        (*-1 AUX-MOD *R)    ;if there is a modal auxiliary (AUX-MOD) to the left (*-1) and if to the right (*R) of it
        (NOT *R V))         ;there is no verb preceeding the word itself (NOT *R V).
```

In the last example, the word *inte* gets @ERR because of its placement after the finite verb in a subordinate clause.[4] The rule illustrates that you can **link several conditions**. The four conditions in the chain make three pairs of linked conditions, the first pair hooked together by *R, the second pair by R+1, and the third pair by *R. The sets in the rule are: ADV-CLAUSAL = clausal adverb, covering a number of common adverbs (e.g. *inte*, *aldrig*, *alltid*) used typically as "satsadverbial" (rather than as "särskilda satsadverbial"); SC = subordinating conjunction; NP-HEAD = nominal phrase head, e.g. N; V-FIN = finite verb; V = verb.

Example: *Söndagens lopp bevisade också att det spelar inte*(@ERR) *någon roll hur väl förberedd man är.*

```
Error detection rule (highly simplified):
(@w =s! (@ERR)              ;For a word (@w), select (s=!) the error tag (@ERR),
        (0 ADV-CLAUSAL)     ;if the word itself is an ADV-CLAUSAL (i.e. belongs to this set), and
        (*-1C SC *R)        ;if there is a SC to the left (*-1) and if to the right (*R) of the SC
        (*RC NP-HEAD R+1)   ;there is a NP-HEAD and the next word to the right (R+1) from the NP-HEAD
        (R+1C V-FIN *R)     ;is a V-FIN and if to the right (*R) of the V-FIN
        (NOT *R V)          ;there is no V preceeding the word itself (NOT *R V), and
        (-1C V-FIN))        ;if the next word to the left from the word itself is V-FIN (-1C V-FIN).
```

In more complex cases there are several chains of linked conditions, both leftwards and rightwards from the target, and there may also be any number of conditions on the words in absolute positions.

It is generally assumed, and it would seem to be an uncontroversial point, that grammar error detection has to be based on syntactic parsing, not necessarily of whole sentences but at least of those parts where errors are anticipated by the system – for instance, in order to be able to find noun phrase internal errors, the system would first have to parse noun phrases. (For systems that build on

some measure of syntactic parsing see e.g. Sågvall Hein 1998, Knutsson 1998, Cooper 1998, Cornu & al. 1996, Bustamante & León 1996.) It is therefore a noteworthy feature of the Grammatifix error detection system that it does **not** build on the output of a syntactic parser. What takes the place of syntactic parsing as such are the context conditions in the @ERR selection rules; in a way, each rule does its own syntactic analysis of a specific sequence of elements and, typically, of the context where it occurs. This is perhaps a burdensome way of writing error detection rules, but, on the other hand, we are saved the trouble of working on parsing rules and their relaxations (cf. Bustamante & León 1996). Anyhow, the conditions for the avoidance of false alarms often being pattern-specific (cf. chapter 3), it is convenient to have pattern-specific error detection rules where to incorporate such conditions. The more local a phenomenon is, the easier it is to control with CG rules.

2.3. Grammar errors and disambiguation

The relationship between disambiguation and grammar error detection is intricate. On the one hand, it is obvious that disambiguation is a prerequisite for any effort at precise error detection. On the other hand, a grammar error may disturb the disambiguation, with either a disambiguation error or remaining ambiguity as a consequence, and this in turn may disturb the error detection.

This section illustrates the methods we use in Grammatifix in order to overcome effects of disambiguation disturbances caused by grammar errors. You can either take the disambiguator's disturbed output as it is and write error detection rules based on that (methods 1 and 2 below), or you can make changes in the disambiguation rules as used by the error detector (method 3). Considering the kinds of Grammatifix rules involved we can speak of three methods: (1) word-form-specific @ERR rules, (2) @ERR rules for ambiguous words, and (3) adjustment of the disambiguation rules. The methods are illustrated in turn below.

Word-form-specific @ERR rules. In the following example (used earlier in chapter 2.2), the correct analysis of the word *ett* would be DET, so in the Grammatifix analysis shown below we have a disambiguation error: PRON instead of the intended DET.

Example: *Ett*(@ERR) *högtrycksrygg förskjuts norrut.*

Lexical analysis of *ett*:
"ett" <NUM/ART> <ID> **DET** NEU INDEF SG NOM ;Correct analysis in **modifier** use.
"ett" <NUM> **PRON** NEU INDEF SG NOM ;Correct analysis in **head** use.

Grammatifix analysis:
"<ett>"
"*ett" <**c> <NUM> **PRON** NEU INDEF SG NOM @ERR
"<högtrycksrygg>"
"högtrycksrygg" N UTR INDEF SG NOM @OK

The grammar error in the above example is – using the grammatically proper terms to describe it – that a neuter **determiner** (DET) is combined with an utrum noun. In the @ERR rule we can not describe the error in those terms, however, because the disambiguator discards the DET reading. What we have here instead is an @ERR rule with the word-form domain "<ett>" (for the rule formulation see chapter 2.2). Grammatifix tolerates the disambiguation error (PRON) simply by ignoring it. Word-form-specific rules are used especially with many common determiners.

Word-form-specific rule can also be formulated so as to cover a set of word forms. In this case the domain of the rule is @w, and the set is used in e.g. the target position (0). For example, in the formulation (@w =s! (@ERR) (0 POSS-UTR) ...), the set POSS-UTR covers utrum forms of possessive determiners, e.g. *sin*. In the sentence *Han har sin*(@ERR) *företag att tänka på*, the disambiguator leaves *sin* three-way ambiguous (DET|PRON|ABBReviation). The ambiguity does not disturb the error detection because the @ERR rule refers to the set POSS-UTR, i.e. ultimately to the word form *sin* in this case.

@ERR rules for ambiguous words. When developing the @ERR rules we noticed cases where words remained ambiguous in some systematic way in certain targeted patterns, and rules were then formulated so as to accept the ambiguity. The following example illustrates the idea. When combined with certain verbs (e.g. *uppges* below) and not preceeded by *ha*, the word (*orsakat* below) whose correct analysis would be supine remains ambiguous between supine and past participle (<PCP2>, with A as part-of-speech tag). (For the message cf. note 3.)

Example: *Slarv uppges orsakat*(@ERR) *branden*

Grammatifix analysis:
"<slarv>"
    "*slarv" <**c> N NEU INDEF SG NOM @OK
"<uppges>"
    "uppge" V PASS PRES @OK
"<orsakat>"
    "orsaka" V ACT **SUPINE** @ERR
    "orsaka" **<PCP2>** **A** NEU INDEF SG NOM @ERR
"<branden>"
    "brand" N UTR DEF SG NOM @OK

What we in this case want to refer to in the @ERR rule is precisely the ambiguity of the target word. This is done with the description (@w =s! (@ERR) (0C SUPINE/PCP2) (0 SUPINE) ...), where the sets are (SUPINE/PCP2 SUPINE <PCP2>) and (SUPINE SUPINE). According to the condition (0C SUPINE/PCP2) the target word must have a SUPINE reading or a <PCP2> reading or both, and according to the condition (0 SUPINE) it must have at least a SUPINE reading. This excludes words that are disambiguated unambiguously as <PCP2>, e.g. premodifiers of nouns. The above target description accepts also words that are unambiguously SUPINE (e.g. *skrivit*).

Adjustment of the disambiguation rules. It was noted in section 2.1 that the disambiguation rules of SWECG have been adopted to a large extent as such in Grammatifix, but, importantly, there are differences. (At present there are some 50 points of difference; if it was not for the two methods discussed above, there would have to be many more.) The most important ones of the differences involve the following scenario. Using the original SWECG disambiguation rules we noticed that certain common open-class words, lexically ambiguous in some systematic way, regularily lost their intended reading in a certain error pattern where the intended reading would have been needed for the @ERR rule to apply. Disambiguation rules were then adjusted in Grammatifix for the recovery of the needed reading. An illustration follows.

It was noticed that common indefinite adjective forms (e.g. *kall* "cold") with a competing indefinite noun reading (*kall* "vocation") regularily lost their adjective reading in the error pattern GEN + A-INDEF(@ERR) + N-INDEF. In the @ERR rule for this pattern, the target word is described as (@w =s! (@ERR) (0C A-INDEF) ...). The rule detected the error e.g. in *Hennes vacker*(@ERR) *hand*, where vacker is A-INDEF, but in the following example the rule did not detect the error because the disambiguator selected the N reading of *kall* instead of the A reading presupposed by the rule.

Example: *Hennes kall hand*

Original SWECG disambiguation:
"<*hennes>"
    "hon" <**c> <PERS-SG3> DET UTR DEF SG GEN
"<kall>"
    "kall" N NEU INDEF SG/PL NOM
"<hand>"
    "hand" N UTR INDEF SG NOM

In SWECG the N analysis above is understandable because *Hennes* + N-INDEF does, whereas *Hennes* + A-INDEF does not, make grammatical sense. In Grammatifix, in order to detect the error, we did the following. First, we defined a set, KALL-ETC, covering words that are ambiguous between A and N in the same way as *kall*, e.g. *besk*, *briljant*, *kall*, *intern*, *sval*, all in all some 60 common adjectives. Then (with due attention to additional details), we wrote a rule specifically for the disambiguation of the KALL-ETC words as A in the context (-1C GEN) (1C N-INDEF). The same @ERR rule that detected the error in *Hennes vacker hand*, now detected the error also in *Hennes kall hand*, as shown below.

Grammatifix analysis:
"<hennes>"
    "*hon" <**c> <PERS-SG3> DET UTR DEF SG GEN @OK
"<kall>"
    "kall" A UTR INDEF SG NOM **@ERR**
"<hand>"
    "hand" N UTR INDEF SG NOM @OK

Using the methods illustrated above we have come to grips with a number of disambiguation disturbances caused by grammar errors, but we are also aware that there are many cases that we have not tackled yet. Some amount of disambiguation errors, be they due to grammar errors or other factors, will always remain a feature of the output of a computational analyser.

3. Notes on the process of refining the error detection rules

The challenge for an error detector is not only to detect errors but also to avoid false alarms. This chapter describes work done in Grammatifix on the avoidance of false alarms.

The issue can be introduced by way of a two-point example. (1) You want to detect the error in the phrase *vilkas diskontinuerliga stavningen*(@ERR), so you write the rule (@w =s! (@ERR) (0 N-DEF) (-2 GEN) (-1 A-DEF)), presented in chapter 2.1. (2) You become aware of cases where you want to **avoid false alarms**, e.g. *Strindbergs Röda rummet*(@OK) and *Dostojevskijs berömda Idioten*(@OK), so you **add conditions** to the @ERR rule to the effect that it does not apply in these cases. The two simple conditions added to the above rule due to these two cases are (NOT -1 CAP) and (NOT 0 CAP), respectively. The set CAP refers to words written with an initial capital letter, e.g. *Röda* and *Idioten*. It is not feasible to list all proper names (e.g. titles of literary works).

The two points in the above introduction correspond to two stages in the process of developing the @ERR rules: (1) constructing a set of basic, rather unconstrained rules, and, based on corpus testing, (2) refining the basic rules. The basic rules were "rather unconstrained" in the sense that they might flag (almost) all instances of a potential error pattern – say, all instances of a noun in definite form preceeded by a genitive, or all instances of *detta* in front of an utrum noun. It was obvious even prior to testing that, of those instances, quite a number would be false alarms. False alarm cases are often so marginal structurally that they easily escape the rule writer's intuitive attention. The purpose of corpus testing was to bring false alarm cases more effectively into our attention, the task then being to refine the rules so that false alarms be eliminated.

The main corpus we used for the purpose explained above was a 1.6 million word collection of published texts, mainly from newspapers and periodicals. (The corpus was compiled by Fredrik Westerlund.) The testing procedure was as follows: the corpus was divided into five parts, and with each part we ran through the same three steps as described below for part 1, except that after part 5 there was no "next part of the corpus" to apply the refined rules to.

The basic rule set applied to part 1 of the corpus
- Each @ERR alarm studied: good or false?
- **False alarms eliminated as far as reasonable**
- Result: a refined rule set, applied to the next part of the corpus

The main point here is that we eliminated false alarms, for examples see below. The reason for treating the corpus in parts was that we wanted to verify that the precision of the rules (ratio of good alarms to all alarms) was improving after each round of rule refinements. It may be noted that when the basic rule set, prior to any refinements, was applied to part 1 of the corpus, the precision was 38%. The precision of the current rule set is reported on in chapter 4.

Below are given some examples (1–9) where a Swedish error detector, if not careful enough in assuming NP internal error patterns, might be tempted into giving false alarms. As for Grammatifix, this is a small sample of cases where the unconstrained rules initially gave a false @ERR alarm but where the refined rules now give @OK (no false alarm). Some notes follow. What these examples signify is an atomistic kind of work process, i.e. cases to be taken into account individually.

(1) *Sveriges* ***Televisions Antikrundan***(@OK) *har slagit ...*
(2) *... Carlos Menem beordrade i* ***fredags flottan***(@OK) *att avföra ...*
(3) *... till* ***dess barnet***(@OK) *fyller 18 år.*
(4) *... har ett* ***slags konstnärlig***(@OK) *frihet att ...*
(5) *... som inte tyckte om sin före* ***detta***(@OK) ***flickväns*** *nye man.*
(6) *... presenterat* ***en***(@OK) ***handfull***(@OK) ***program*** *med samma ...*
(7) ***Obetald***(@OK) ***omslags-*** *eller sällskapsflicka ...*
(8) *Walters far är gammelkommunisten för* ***vilken demokratin***(@OK) *börjar utanför dörren.*
(9) *Från början var* ***det stora problem***(@OK) *att få i Johan tillräckligt med mat.*

Notes on (1–9):

- The potential false alarm sources in (1–9) are: a noun in definite form is preceeded by a a word in genitive form (1–3); an adjective in indefinite form is preceeded by a word in genitive form (4); an utrum noun is preceeded by *detta* (5); a neuter noun is preceeded by *en* and an utrum adjective (*handfull*) (6); a neuter noun is preceeded by an adjective in utrum form (7); a noun in definite form is preceeded by *vilken* (8); a noun in indefinite form is preceeded by *det* and a potential definite form of an adjective (9).
- Example (2). The best way to avoid a false alarm in (2) is probably to treat *i freadgs* as a fixed syntagm, i.e. *"i_fredags"* ADV, cf. the notes on preprocessing in chapter 2.1.
- Example (3). This example can be used to illustrate a kind of chain reaction that may occur as a consequence of adding a condition to a rule. First, a general rule for GEN + N-DEF(@ERR) detects the error e.g. in *Onsdagens finalen*(@ERR) *visas i TV.* In that general rule we have added the condition (NOT -1 DESS) in order to avoid a false alarm in (3). Then, in order to detect the error e.g. in *Dess framtiden*(@ERR) *är osäker*, we have written a rule specifically for *dess* + N-DEF(@ERR), and (only) in that rule we use the condition (NOT -2 TILL), again in order to avoid a false alarm in (3). This kind of chains may sometimes be the only way of achieving the desired pattern- or word-specific effects.
- Example (6). The word *handfull*, classified as an adjective in SWETWOL, was excluded from the adjective slot in the relevant @ERR rules for the avoidance of false alarms in cases like (6) (quite frequent in texts). A syntactically more perceptive solution would have been to provide *handfull* with a noun reading (cf. *ett antal program*).
- Example (8). In (8) and (9), more than in the previous examples, the false alarm conditions are clause-structurally oriented. (8) can be compared with the following sentence where Grammatifix properly detects the error: *Saab var riktmärket för* ***vilken bilmodellen***(@ERR) *var och en skulle ha.* In this sentence the sequence *vilken* + N-DEF is followed by a NP boundary (*var och en*), whereas this is not the case in (8), this distinction taken into account in the rule conditions.
- Example (9). The single most problematic word for Swedish (agreement) error detection cum false alarm avoidance is *det*. This is because of the many uses of *det* as an independent clausal element particularily in a position after the finite verb. Let us consider one of the potential error patterns, viz. the one exemplified by *det stora problem* in (8) (*det* + A-DEF/INDEF-SG/PL + N-NEU-INDEF-SG/PL). The relatively **safest** clausal position for assuming an error, i.e. the position with the least chances for false alarms, is the initial pre-finite-verb position (= PRE-FV), e.g. *Det stora*

problem(@ERR) *har lösts*. The **next safest** position for assuming an error is the post-non-finite-verb position (= POST-NONFV), e.g. *Hon har löst det stora problem*(@ERR). More conditions are pertinent in POST-NONFV than in PRE-FV. You need to check that the non-finite verb is not ditransitive, cf. *Ni har vållat det stora problem*(@OK), and also that no relative clause follows, cf. *löst det stora problem*(@OK) *som ni funderat på*. The **least safe** position for assuming an error is the post-finite-verb position (= POST-FV). In addition to the possibilities in POST-NONFV, in POST-FV you have to consider that *det* may function as subject, e.g. *Här framkallar det stora problem*(@OK), or as formal subject, e.g. (8) and *Här finns det stora problem*(@ERR). Because such POST-FV uses of *det* are so common, and in any case much more frequent than erroneous uses, it would seem to be motivated to prevent the @ERR rules here concerned from applying in POST-FV. However, not all POST-FV contexts are equal. For instance, it is relatively safe to assume an error if the finite verb is an auxiliary and a non-finite verb follows, e.g. *Nu har det stora problem*(@ERR) *lösts*. This is more or less as far as we have come with the description of the *det* pattern here discussed. – A clause construction where Grammatifix at present misses an error is exemplified by *Hennes assistanter löste det stora problem*(@OK, missed error). Crucial factors here are that the clause-initial constituent is a (non-adverbial) noun phrase (*Hennes assistanter*), and that the finite verb is monotransitive (*löste*). On the basis of that information it would be motivated to flag *problem* as @ERR in the above example, but, as noted, this has not yet been worked into the Grammatifix rule set.

Users soon get tired of a language checker that makes a lot of false alarms. In a practical grammar checker it is therefore motivated to make false alarm avoidance a priority even at the expense of errors being missed – but there is a limit, of course. It was noted above that, in the corpus used for rule refinement purposes, we tried to eliminate false alarms "as far as reasonable". This eludes exact definition, but the flexible idea is that we do not insist on conditions for false alarm avoidance if they would unduly compromise the rule's error detection power in unintended contexts. To illustrate, below are given two examples of a problematic construction, ellipsis of the verb gapping type. Grammatifix makes here false alarms: it believes that *de andra medlemmar* is a phrase with a definiteness form error, and that *den andra hormonet* is a phrase with a gender agreement error.

Nitton av dem ska ha varit medlemmar av Umma-partiet och de andra medlemmar(@ERR, false alarm) *av det förbjudna arabsocialistiska Baathpartiet.*
Ena kammaren innehåller destillerat vatten, och den(@ERR, false alarm) *andra hormonet.*

It would be possible to eliminate many of the false alarms due to ellipsis, e.g. by using conditions that refer to a coordinator or a comma in the left context. However, such conditions would not be precise enough – they would prevent the rules from making valid detections in many contexts that have nothing to do with ellipsis. False alarms due to ellipsis are not reasonably to be avoided, the ultimate reason being that ellipsis is too elusive for us to identify exclusively.

A grammar checker is a compromise between error detection and false alarm avoidance. One way of describing such a compromise is to provide test results on precision and recall (see the final chapter). In the end, the user is the arbiter of whether the compromise is acceptable or not.

4. Performance tests

Introduction. For this presentation, we tested our set of grammar error detection rules (i.e. @ERR selection rules) for overall precision and recall with texts new to the system. Precision and recall can here be defined as follows (cf. Bernth 1997:159, Paggio & Music 1999:278).

Precision: the ratio 'good alarms / all alarms'
Recall: the ratio 'detected errors / all errors'

Precision is a measure of how good the checker is at avoiding false (unintended, irrelevant) alarms, and recall is a measure of how good the checker is at identifying the errors in a text – the higher the recall and the precision, the better. As the term 'error' is used here it covers, in addition to undisputable grammar violations (e.g. the verb chain *kan* + *blir* in *Då kan bland annat så kallade utbildningskonton blir aktuella.*), also constructions targeted by Grammatifix which, acceptable to some, are not regarded as impeccable by everybody (e.g. the verb chain *kommer* + *sätta* in *De kommer sätta stenhårt tryck på oss.*). We are here concerned with grammatical errors, so nothing will be said about e.g. spelling errors (the concern of a spell-checker) and writing convention errors (the concern of a separate set of Grammatifix rules, cf. chapter 1).

There is no standard for how the performance of a (Swedish) grammar error detector should be evaluated. One issue here is the kind of test data used. Research groups often seem to use their error corpuses, i.e. collections of sentences containing errors of the types the system is concerned with (cf. Domeij & Knutsson 1999, Paggio & Music 1998, Cornu & al. 1996, Bolioli & al. 1992). In the tests here reported we used running newspaper text. Results based on such data are not comparable to results based on a collection of errors. Especially, running newspaper text, with a high proportion of grammatically correct sentences, puts precision to a hard test. – A fundamental kind of problem is also that there are no agreed-upon criteria for what should count as a grammar error or as a good alarm (in border-line cases). A further factor that would complicate comparative evaluation is the variation in the error types targeted by different systems, e.g. where one system tries to detect only easiest-to-identify errors while another system tries to detect also more-difficult-to-identify errors.

In anticipation of more carefully planned and documented test schemes – schemes that would address open issues such as the ones noted above – we present below our precision and recall tests.

Precision. The test data is a 1,000,504 word extract from the Swedish newspaper *Göteborgs-Posten* 1998. Of the grammatical categories that Grammatifix has checks on (presented in Arppe & al. 1999, listed also in Arppe in this volume) two were excluded from the test. These two are 'no verb' (e.g. *Ungefär som en kansler.)* and 'no finite verb' (e.g. *Göra independentfilm till exempel.*). Grammatifix points out these properties of sentences, but in almost all cases no error is involved. These properties are rather frequent and easy to detect reliably.[5]

The test data was analysed by the @ERR selection rules (in Unix); each alarm was studied as to whether it was good or false; the numbers of good and false alarms were used for calculating the precision rate. The result is given below, both as a percentage and as absolute figures.

Precision of Grammatifix in a 1,000,504 word extract from *Göteborgs-Posten* 1998:

Good alarms	False alarms	**Precision**
374	160	**70%** (374 / 374 + 160)

Is 70% a good or a bad overall result? It is hard to say as we have not found similar test reports to compare with. What the result would be with other types of text remains an open question, too.

Perhaps the most relevant point to make concerning the 374 good alarms is, simply, that simple grammar errors do occur even in published texts produced by native writers. We may illustrate with some noun phrase internal agreement errors (1–9 below) detected by Grammatifix in the test data.[6] These are typical agreement violations in the sense that each of them involves only one agreement feature, gender in (1–3), number in (4–6), and definiteness in (7–9).

(1) ... *som beskriver världen utifrån en*(@ERR) *annan*(@ERR) *paradigm än till exempel Newton och Descartes.*
(2) ... *blev i stället en stort*(@ERR) *besvikelse för Pernilla Wiberg.*
(3) *Det är ju utlänningar i nästan varenda*(@ERR) *bås nuförtiden.*
(4) *Men på de*(@ERR) *mest framskjutna platsen i montrarna står det mer rustika porslinet ...*
(5) *Mir har under det*(@ERR) *senaste åren drabbats av flera svåra olyckor.*
(6) *Polisen gjorde färre fordonskontroll*(@ERR) *förra året ...*
(7) *I går häktades den 32-årige stockholmare*(@ERR) *vid Stockholms tingsrätt, misstänkt för grovt häleri.*
(8) ... *inte hade lyckats uppnå samma ekonomisk*(@ERR) *utveckling.*
(9) ... *utan att någondera familj*(@ERR) *förstod varför katten blev allt fetare.*

The general-level sources for the 160 false alarms in the test data are: lexical gaps or errors (46 out of 160); disambiguation errors (18); not accurate enough @ERR selection rules (96). Some of these false alarms will be easy to eliminate, the easiest ones being among those due to a lexical source, e.g. the word *partnerskap*, treated as SG in the current SWETWOL (and in SAOL), could be tagged as SG/PL for the elimination of the false number agreement alarm in *Humankapitalet skyddas lämpligen genom nya*(@ERR) *partnerskap vid företagande (...)*. On the other hand, some of the false alarms are such that we do not consider it reasonable even to try to avoid them, e.g. the ellipsis-induced (cf. chapter 3) false number agreeement alarm in *Antal toaletter: tre, varav två tjej(@ERR) och en kill.*

Recall. The test data is a 87,713 word extract from *Göteborgs-Posten* 1998. Two linguists (the present author and Eva Orava, also at Lingsoft) read the extract, marked all the grammar errors they found, and discussed problem cases. What they ended up with was 135 grammar errors distributed over different categories as follows: agreement errors (31), most of them NP-internal (22), the rest (9) involving complements, postmodifiers, and anaphoric pronouns; verb form compatibility errors (28), especially violations of verb chain internal constraints; preposition errors (26); missing or superfluous endings (21), e.g. genitive, passive, or adverb endings; compounds written as separate words (8); sentence structure errors (8); word order errors (3); others (10).

Of the 135 grammar errors found by the linguists in the test data, 55 belong to the categories targeted by Grammatifix[7]. Any computational grammar checker is, of course, only a partial grammar checker; no current systems have anything like comprehensive checking of, say, sentence structure, anaphoric pronoun agreement, missing endings, and even preposition use (in other than some types of fixed phrases perhaps). Now, is it more to the point to calculate recall in relation to 'all errors in all the error categories', or in relation to 'all errors in the error categories targeted by the system'? The results of both calculations are given below. Recall in relation to the targeted categories is an overall measure of how well the rule set does what it tries to do.

Recall of Grammatifix in a 87,713 word extract from *Göteborgs-Posten* 1998:

	All errors in text	Detected errors	**Recall**
Targetted error categories:	55	47	**85%** (47/55)
All error categories:	135	47	35 % (47/135)

The general-level sources for missed errors in the targeted categories (in the test data, 55-47 = 8) are the same as those for false alarms, i.e. lexicon, disambiguation, and @ERR rules (cf. above). 'Not accurate enough' @ERR rules means here that errors are missed due to overly prohibitive conditions for the avoidance of false alarms (in the test data, 5 of the 8 misses). One of the problems associated with the recall test here reported is the small size of the test data.

Notes

[1] For spell-checking Lingsoft has a separate program (Orthografix).

[2] The part-of-speech tags referred to in this paper are: A = adjective, ADV = adverb, DET = determiner, N = noun, PRON = pronoun, SC = subordinating conjunction, V = verb. Nominal minor feature tags include: DEF = definite, GEN = genitive, INDEF = indefinite, NOM = nominative, NEU = neutrum, PL = plural, SG = singular, UTR = utrum. Other tags and set names, if not transparent, will be explained when referred to.

[3] The construction 'modal auxiliary + supine without *ha*' (e.g. *skulle ... skrivits*) is not regarded by everybody as recommendable in polished style. The Grammatifix message is that, in polished style, a modal auxiliary is combined rather with *ha* + supine than with supine alone. Cf. Wellander (1973:139).

[4] There are differences between subordinate clauses as to the usability of the word order 'finite verb + clausal adverb' (Teleman & al. 1999:537-9). In formal written Swedish, however, the order 'clausal adverb + finite verb' can be regarded as recommendable in all types of subordinate clauses. Cf. Reuter (1996:8), Åberg (1995:34), *Dagens Nyheters Skrivregler* (1997:30).

[5] In the precision test data, the rules made 7145 'no verb' alarms, and 321 'no finite verb' alarms. The latter ones were studied more in detail: of the 321 alarms, 312 were good, i.e. cases where the sentence included a non-finite verb but

no finite verb. The precision for this alarm type was 97% (312/321). A few real errors detected by the rules (e.g. *Pulsen bli för kraftig*) were ignored in the precision test as a consequence of excluding the alarm types.

[6] Of the 374 good alarms, 134 were concerned with noun phrase internal agreement. The other big group was verb form compatibility with 176 good alarms; 99 of these were supines without *ha*, e.g. *kunde den fått*(@ERR) (cf. note 3).

[7] Among these 55, the two big groups were noun phrase internal agreement violations (22) and verb chain internal compatibility violations (21).

References

Arppe, Antti (this volume). Developing a grammar checker for Swedish.

Arppe, Antti, Juhani Birn, and Fredrik Westerlund 1999. Lingsoft's Swedish Grammar Checker. http://www.lingsoft.fi/doc/swegc/.

Bernth, Arendse 1997. Easy English: A Tool for Improving Document Quality. *Proceedings of the Fifth Conference on Applied Language Processing*, Washington, 159–165.

Bolioli, Andrea, Luca Dini, and Giovanni Malnati 1992. JDII: Parsing Italian with a Robust Constraint Grammar. *Proceedings of the 15th International Conference on Computational Linguistics*, Nantes, 1003–1007.

Bustamante, Flora Ramiréz and Fernando Sánchez León 1996. GramCheck: A Grammar and Style Checker. *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, 175–181.

Birn, Juhani 1998. Swedish Constraint Grammar: A Short Presentation. http://www.lingsoft.fi/doc/swecg/.

Cooper, Robin 1998. Finite state grammar for finding grammatical errors in Swedish text. http://www.ling.gu.se/~sylvana/FSG/.

Cornu, Etienne, Natalie Kubler, Franck Bodmer, Francois Grosjean, Lysiane Grosjean, Nicolas Lévy, Cornelia Tschichold, and Corinne Tschumi 1996. Prototype of a second language writing tool for French speakers writing in English. *Natural Language Engineering* 2, 211–238. Cambridge University Press.

Domeij, Rickard and Ola Knutsson 1999. Granska - ett effektivt hybridsystem för svensk grammatikkontroll. http://www.nada.kth.se/theory/projects/granska/rapporter/nodalidaabstrakt.html.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila (eds.) 1995. *Constraint Grammar. A Language-Independent System for Parsing Unrestricted Text.* Berlin and New York: Mouton de Gruyter.

Knutsson, Ola 1999. Granskas regelspråk. At: http://www.nada.kth.se/theory/projects/granska/.

Paggio, Patrizia and Bradley Music 1998. Evaluation in the Scarrie project. *Proceedings of the First International Conference on Language Resources & Evaluation*, Granada, 277–282.

Reuter, Mikael 1996. *Reuters rutor 2.* Esbo: Schildts.

SAOL, *Svenska Akademiens ordlista över svenska språket.* Tolfte upplagan. 1998. Norstedts.

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

Teleman, Ulf, Staffan Hellberg, and Erik Andersson 1999. *Svenska Akademiens grammatik 4.* Norstedts.

Wellander, Erik 1973. *Riktig svenska.* Stockholm: Esselte studium.

Åberg, Gösta 1995. *Hur ska det heta? Tidens lilla språkriktighetslexikon.* Stockholm: Tidens förlag.

Pivot alignment

Lars Borin
Department of Linguistics
Uppsala University
Lars.Borin@ling.uu.se

Abstract

Word alignment of parallel texts is typically carried out using many kinds of knowledge, or information sources, in concert, i.e., it is profitably viewed as a kind of cooperative process, where e.g. distribution, string similarity, cooccurrence statistics, and other information sources are used together. We investigate a novel such information source in this paper, namely the use of a third language as a 'pivot' to increase alignment recall, hence the name *pivot alignment*. The results of the preliminary experiments reported here indicate that pivot alignment increases word alignment recall, without sacrificing precision. We conclude that the method is well worth exploring further, by examining more languages and language combinations.

1 Introduction

Parallel texts aligned on the word level have a number of potential uses. Given suitable browsing and search tools, linguists can use aligned parallel corpora in the same way that they already use monolingual corpora, i.e. as a rich source of authentic language data, in this case data on translation equivalence (see, e.g., Olsson & Borin forthcoming). Bilingual lexicography, translator training, and foreign language instruction all stand to benefit from the use of such corpora. In computational linguistics, the application which springs to mind first is the automatic or semi-automatic extraction of translation equivalents for machine translation systems from word-aligned parallel texts, but there are also possible applications in the fields of computer-assisted language learning and cross-lingual information retrieval.

The ETAP project is a parallel translation corpus project funded by the Bank of Sweden Tercentenary Foundation. The aim of this project is to create an annotated and word-aligned multilingual translation corpus, which will be used as the basis for the development of methods and tools for automatic extraction of translation equivalents on the word and phrase levels (see Borin forthcoming a).

2 Word alignment as a cooperative process

Sentence alignment is a fairly well-understood problem, with state-of-the-art sentence alignment algorithms routinely achieving accuracies close to a hundred percent,[1] even

without the use of language-specific information. The best word alignment systems, on the other hand, typically achieve a recall in the 25 to 45 percent range in the language-independent case (but with high precision, typically over ninety percent).[2]

In common with many other nontrivial linguistic tasks, the decisions of which words to link up with each other, i.e. the ability to make correct word alignments, seem to draw on many different knowledge sources simultaneously. The word alignment system that we use in the ETAP project, the UWA (Uppsala Word Aligner; see Tiedemann this volume), uses several kinds of information in an iterative word alignment process, where a text-specific translation dictionary is accumulated, and aligned units are removed after each step. The following kinds of information are used to align words (this is an extremely simplified account of how the UWA works; see Tiedemann this volume for details).

- single-word 'sentences', which may be the result of previous removals of words from multi-word sentences[3]
- identical and highly similar words
- distributionally similar words

Additionaly, language-internal cooccurrence statistics are used to find multi-word units ('phrases') in both languages, which can then be aligned in the same way as single words, while lowercasing and stemming reduce the number of types, thus increasing the average type frequency, making statistical methods more effective.

Thus, we see that the UWA uses many kinds of knowledge to achieve its objective. In the same spirit, we have explored the possibilities of combining word alignment and part-of-speech (POS) tagging (Borin forthcoming c), as well as combining different POS taggers using linguistically motivated rules, so that the combination achieves greater accuracy than the best individual tagger (Borin forthcoming d).

All this have led us to a view of word (and phrase and sentence) alignment, and also POS tagging, as a *cooperative process*, where many independent 'experts', using various kinds of information sources, access and modify the same, increasingly richer linguistic representation, performing POS tagging, alignment, and possibly other kinds of linguistic analysis and annotation as well, utilizing the relevant information that other experts have left there.[4]

We already know that distributional parallelism, language-internal and cross-language coocurrence, string similarity (also both between and within languages), and part of speech are useful information sources for word alignment (Tiedemann 1998, this volume, forthcoming; Melamed 1995, 1998; Borin forthcoming c).

The view of word alignment as being achieved by the use of many (mutually independent) kinds of knowledge in concert naturally makes one look for additional such kinds of knowledge, information sources that could be used to further improve word alignment. This paper discusses one such source which to the best of my knowledge has not been considered earlier, namely the use of a third language in the alignment process. Perhaps the reason that it has not been considered earlier is that it is possible only with *multi*lingual parallel corpora, and—for obvious reasons—not with *bi*lingual

corpora, which has been the kind of parallel corpus that has received most attention from researchers in the field.

3 Pivot alignment

Since the third language acts as, as it were, a pivot for the alignment of the two other languages, we refer to the method as *pivot alignment*, and it works as follows, with three languages, e.g. Swedish (SE), Polish (PL) and Serbian-Bosnian-Croatian (SBC), where the aim is to align Swedish with the other two languages on the word level.

1. Perform the pairwise alignments SE→PL, SE→SBC, PL→SBC, and SBC→PL;[5]
2. Check whether there exist aligned words on the indirect 'alignment path' SE→SBC→PL, which are not on the direct path SE→PL. If there are, add them to the SE→PL alignments;
3. Do the same for the indirect path SE→PL→SBC and the direct path SE→SBC.

In order for this procedure to work, we must believe that

1. there will be differences in the SE→PL and SE→SBC alignments, and
2. that these differences will 'survive' the PL→SBC and SBC→PL alignments.[6]

In other words, the indirect alignment path must *add* information to the one yielded by the direct path. If we can conceive of some plausible reason for this to happen, we may believe in the first hypothesis. One such good reason could be the fact that, as mentioned earlier, the word alignment system used, UWA, utilizes several kinds of information to align the words in the two texts. Thus it is fully conceivable, e.g., that distributional information will provide one of the links and word similarity the other in a three-language path, such as SE→PL→SBC, while synonymy or polysemy (i.e., distributional differences) prevents the first link to be made on the direct path SE→SBC. Intuitively, this is perhaps the most likely situation in this particular example, since Polish and Serbian-Bosnian-Croatian are fairly closely related Slavic languages which share many easily recognizable cognates, while both are much more remotely related to Swedish.

4 An experiment with pivot alignment

To test these hypotheses, we performed a small experiment with pivot alignment, as follows.

1. The ETAP IVT1 corpus was used for the experiment. This is a five-language parallel translation corpus consisting of text from the Swedish newspaper for immigrants (*Invandrartidningen*; the English version is called *News and Views*). Swedish is the source language, and the other four languages are English (EN), Polish, Serbian-Bosnian-Croatian and Spanish (ES). The IVT1 corpus has roughly 100,000 words of text in each language;

2. The PLUG Link Annotator (Merkel 1999; Merkel *et al.* forthcoming), was used to produce evaluation standards for the following alignment directions: SE→PL, SE→SBC, PL→SBC, SBC→PL in one group, and SE→EN, SE→ES, EN→ES, ES→EN in the other. A total of 500 words were sampled randomly from the full Swedish source text, and the standards with Swedish as the source were made manually by me from this sample. The target units of these standards were then used as the basis for the manual establishment (again by me) of the various target language alignment evaluation standards. Because of null links, misaligned or differently aligned sentences, etc., the size of the evaluation standards varies from 366 to 500 words;

3. In addition to the already word aligned SE→{EN,ES,PL,SBC}, we aligned the other language pairs necessary for the experiment;

4. The word link evaluation tool of the Uplug system, a parallel corpus toolbox of which the UWA is one component (see Tiedemann forthcoming), was used to calculate recall and precision for each language pair (i.e., SE→{EN,ES,PL,SBC}) word alignment. In addition to this, we manually extracted the additional links, if any, that would be found on the indirect path through the third language.

The results of the experiment are shown in Table 1.

languages aligned	*found links*	*links in standard*	*recall*	*correct (C)*	*partly corr. (PC)*	*not corr.*	*precision, correct*	*precision C + PC*
se-sbc	82	429	19.11%	57	17	8	69.51%	90.24%
+ se-pl-sbc	1			1				
=	83		19.35%	58	17	8	69.88%	90.36%
se-pl	57	370	15.41%	37	14	6	64.91%	89.47%
+ se-sbc-pl	4			4				
=	61		16.49%	41	14	6	67.21%	90.16%
se-es	87	454	19.16%	65	14	8	74.71%	90.80%
+ se-en-es	8			7		1		
=	95		20.92%	72	14	9	75.79%	90.53%
se-en	95	442	21.49%	70	14	11	73.68%	88.42%
+ se-es-en	4			2		2		
=	99		22.40%	72	14	13	72.73%	86.87%

Table 1: Pivot alignment experiment results (null links in standard not counted)

The "partly correct" alignments are those where part(s) of a multi-word-unit, but not all of it, have been correctly aligned. As you can see in Table 1, the potentially thorny issue did not arise of how to count a partially correct link added by pivot alignment.

We see that only a few units survived the trip through two languages, but out of those that did, most contributed positively to the total result. SE→ES and SE→PL were the alignments which benefitted most from pivot alignment (through EN and SBC, respectively), while the result was insignificant for SE→SBC and perhaps even slightly detrimental in the case of SE→EN.

5 Discussion

The material examined is fairly small, and it would be fair to say that the results presented above are best treated as suggestive, rather than conclusive. I think we may be said to have made a case for the usefulness of pivot alignment, as it tended to increase the overall recall, without lowering precision. In other words, the links added by pivot alignment tend to be good links.

Several ways suggest themselves in which the research presented here could be extended to see whether the case holds upon closer scrutiny.

In the results, there are differences between languages, even in this small material, but not exactly those that we had expected. Recall that we speculated (at least implicitly) that using a language closely related to the target language as a pivot would be more effective than using a combination of relatively more remote languages. Thus, we would have predicted that SE→PL and SE→SBC would come out on top in Table 1, which obviously was not the case. This could be due to chance, but also to some other factor. There is also the circumstance that English is actually very close to the Romance languages (of which Spanish is one) in its vocabulary, so that we may in fact have been comparing two quite similar cases.

To investigate this, we intend to perform the same kind of experiment with the other possible pivot languages in the IVT1 corpus, still using Swedish as the source, e.g. aligning Swedish and Polish, using Spanish as pivot. In this way, genetic factors should be more clearly discernible. We will also include at least Finnish in future experiments, as a representative of another language family (all the languages in the experiment were Indo-European languages; Finnish is the only non-Indo-European language included in the IVT corpus at present).

The planned experiments where the same language pair will be aligned with different pivot languages will make it possible to investigate whether pivot alignment is 'cumulative', i.e., whether

1. each pivot language contributes positively to the alignment, and
2. different pivot languages contribute different additional alignments.

In this case, we would have, not only pivot alignment in general as an additional 'expert', but each new language in a multilingual parallel corpus could then, potentially, make the annotation of all the other languages in the corpus richer.

The Plug Link Annotator is a very useful tool, without which the experiments described here could not have been carried out. It was originally developed with another goal in mind, however, that of evaluating word alignment systems. Hence, it is not surprising that we found, in the course of our work, that the PLA could be made even more useful for our purposes. Two modifications in particular would facilitate further experiments with pivot alignment, one more trivial and one more fundamental:

1. The sampling procedure should be modified to exclude function words. They tend to have a high text frequency, and thus make up a sizeable part of any random sample. Most of the null links in the experiment reported here resulted from function words, the typical case here being that of personal pronouns in Swedish,

where the equivalent information is normally expressed by person marking on the verb in Polish, Serbian-Bosnian-Croatian and Spanish, and only rarely by a separate pronoun.

2. At the moment, at most one word is sampled in each sentence alignment unit. For our purposes, it would be better if sentences were sampled, instead of words, and that the annotator be allowed to link as many words as desired in the sentence alignment unit of the sampled sentence. This would allow us to follow up on the misaligned source language units, which at present cannot be tracked through the pivot language, because the 'sample' for the pivot language is made up of the correct target words only. As the UWA aligns words only within sentence alignment units, working with sentences instead of words as sampling units would hopefully make it possible to follow up also on incorrect alignments.

A simple approximation in the first case would be to exclude high-frequency items from the sampling, if it is deemed desirable to avoid introducing language-specific information. This is a comparatively simple measure to take, and certainly one that we will take in the next round of experiments with pivot alignment.

The second problem requires for its solution a major redesign of the Plug Link Annotator, which is something that might be worth undertaking in case further experiments confirm the preliminary conclusions reached here.

It would seem that pivot alignment is suited mainly for parallel *translation corpora*, and not for the kind of corpora sometimes called simply parallel corpora, sometimes *comparable corpora*, i.e., corpora, where 'the same kind' (comparable with regard to topic, style, etc.) of text material has been collected in several languages, and mainly statistical (distributional) methods are used to locate equivalent items in the different language versions. It is possible that (a kind of) pivot alignment could be used also with comparable multilingual corpora, and this is certainly an idea worth pursuing.

6 Conclusion

In conclusion, we may say that the results of the experiments presented here are encouraging, although not conclusive. It turned out that the links added by pivot alignment were largely correct links, i.e. pivot alignment could be expected to make a positive contribution in a word alignment system using many independent information sources.

We saw that the sampling procedure and annotating program used could be optimized for this kind of investigation. The results also pointed to natural ways of extending the work reported here, by the investigation of

- more language combinations and more pivot languages, including non-Indo-European ones
- the effect of using two or more pivot languages in parallel
- the possibilities of using (a procedure similar to) pivot alignment also on comparable (parallel non-translation) corpora

Notes

[0]The research reported here was carried out within the ETAP (*Etablering och annotering av parallellkorpus för igenkänning av översättningsekvivalenter*; in English: "Creating and annotating a parallel corpus for the recognition of translation equivalents") project, supported by the Bank of Sweden Tercentenary Foundation as part of the research programme *Translation and Interpreting—a Meeting between Languages and Cultures*. See http://www.translation.su.se/. Leif-Jöran Olsson, who is responsible for systems development in the ETAP project, wrote most of the software which made the experiment reported here possible. I wish to thank the members of the PLUG project (see Ahrenberg *et al.* 1998; Sågvall Hein forthcoming) for generously letting us use the Uplug system, including the Uppsala Word Aligner, and the PLUG Link Annotator.

[1]Although there are still some unresolved issues even in sentence alignment (see McEnery & Oakes 1996; Borin forthcoming b). Our empirical experience shows that its accuracy is dependent upon many factors, such as text type, the quality of the translation, the tokenization algorithm used, etc.

[2]By the *recall* of a word alignment system, we here mean the number of (total or partial) alignments (or links) returned, divided by the number of alignments established in the text pair by a human annotator (i.e., we work with a manually established evaluation standard; see below, and also Merkel 1999; Merkel *et al.* forthcoming), while *precision* is the number of correct alignments returned divided by the total number of returned links. Thus, if the human annotator has established a standard containing 200 links in a text corpus, and the word alignment system returns links for 80 of the *source language words* in the standard, its recall is 40% (80/200). If 74 of those 80 links are correct (according to the standard), the precision becomes 92.5% (74/80). In this paper, we disregard the question of how to count null links—source language words in the standard which explicitly should remain unlinked—when calculating recall and precision, not because it is unimportant, but because we cannot see that it bears directly upon the issues discussed here.

[3]The UWA presupposes a sentence-aligned input corpus, and performs word alignments only within the existing sentence alignment units (thus, if the sentence alignment is wrong, for some reason, the word aligner will not be able to correct it).

[4]Our picture of what the ideal word alignment system would look like has much in common with the "blackboard model", which was once popular in Artificial Intelligence (see, e.g., Patterson 1990).

[5]It may seem strange that we make *both* the PL→SBC *and* the SBC→PL alignments. Intuitively, one would think that the direction would not matter, i.e., that these two alignments would result in the same set of word links. However, we have not checked whether the alignment system used (the Uppsala Word Aligner; see Tiedemann this volume) actually works in this way (this could be the topic of an interesting investigation in its own right). Thus, in order not to introduce a possibly confounding extra variable in the experiment, we decided to treat the alignment as directional (guilty until proven innocent, as it were), and to use both alignments.

[6]Incidentally, the indirect path could be extended with more languages, e.g. Swedish→Polish→English→Spanish, etc., but we have not investigated this possibility.

References

Ahrenberg, L., Merkel, M., Mühlenbock, K., Ridings, D., Sågvall Hein, A. & Tiedemann, J. 1998. Automatic Processing of Parallel Corpora. A Swedish Perspective. Linköping: Electronic University Press.

Borin, L. 1998. Linguistics Isn't Always the Answer: Word Comparison in Computational Linguistics. *NODALIDA '98 Proceedings*. Center for Sprogteknologi & Dept. of General and Applied Linguistics, University of Copenhagen. 140–151.

Borin, L. forthcoming a. The ETAP Project – a Presentation and Status Report. ETAP Technical Report etap-rr-01. Dept. of Linguistics, Uppsala University.

Borin, L. forthcoming b. ...and Never the Twain Shall Meet? *Parallel Corpora, Parallel Worlds*, ed. by Lars Borin. Dept. of Linguistics, Uppsala University.

Borin, L. forthcoming c. Alignment and Tagging. *Parallel Corpora, Parallel Worlds*, ed. by Lars Borin. Dept. of Linguistics, Uppsala University.

Borin, L. forthcoming d. Something Borrowed, Something Blue: Rule-Based Combination of POS Taggers. *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC2000)*, Athens, Greece, 31 May–2 June, 2000.

Melamed, I. D. 1995. Automatic Evaluation and Uniform Filter Cascades for Inducing N-Best Translation Lexicons. *Proceedings of the Third Workshop on Very Large Corpora.* Boston, Massachusetts.

Melamed, I. D. 1998. Word-to-Word Models of Translational Equivalence. IRCS Technical Report #98–08. Dept. of Computer and Information Science, University of Pennsylvania.

Merkel, M. 1999. *Understanding and Enhancing Translation by Parallel Text Processing*. Dept. of Computer and Information Science, Linköping University.

Merkel, M., Andersson, M. & Ahrenberg, L. forthcoming. The PLUG Link Annotator – Interactive Construction of Data from Parallel Corpora. *Parallel Corpora, Parallel Worlds*, ed. by Lars Borin. Dept. of Linguistics, Uppsala University.

McEnery, T. & Oakes, M. 1996. Sentence and Word Alignment in the CRATER Project. *Using Corpora for Language Research. Studies in Honour of Geoffrey Leech*, ed. by Jenny Thomas and Mick Short. London: Longman. 220–230.

Olsson, L.-J. & Borin, L. forthcoming. A Web-Based Tool for Exploring Translation Equivalents on Word and Sentence Level in Multilingual Parallel Corpora. Paper to be presented at the VAKKI Symposium, 12–13 February, 2000, Vaasa University, Finland.

Patterson, D. W. 1990. *Introduction to Artificial Intelligence and Expert Systems.* Englewood Cliffs, N.J.: Prentice-Hall.

Sågvall Hein, A. forthcoming. The PLUG Project: Parallel Corpora in Linköping, Uppsala, Göteborg: Aims and Achievements. *Parallel Corpora, Parallel Worlds*, ed. by Lars Borin. Dept. of Linguistics, Uppsala University.

Tiedemann, J. 1998. Extraction of Translation Equivalents from Parallel Corpora. *NODALIDA '98 Proceedings.* Center for Sprogteknologi & Dept. of General and Applied Linguistics, University of Copenhagen. 120–128.

Tiedemann, J. this volume. Word Alignment Step by Step.

Tiedemann, J. forthcoming. Uplug – a Modular Corpus Tool for Parallel Corpora. *Parallel Corpora, Parallel Worlds*, ed. by Lars Borin. Dept. of Linguistics, Uppsala University.

Granska
- an efficient hybrid system for Swedish grammar checking

Rickard Domeij, Ola Knutsson, Johan Carlberger, Viggo Kann
Nada, KTH, Stockholm
Dept. of Linguistics, Stockholm University
{domeij, knutsson, jfc, viggo}@nada.kth.se

***

Abstract

This article describes how Granska – a surface-oriented system for checking Swedish grammar – is constructed. With the use of special error detection rules, the system can detect and suggest corrections for a number of grammatical errors in Swedish texts. Specifically, we focus on how erroneously split compounds and noun phrase agreement are handled in the rules.

The system combines probabilistic and rule-based methods to achieve high efficiency and robustness. This is a necessary prerequisite for a grammar checker that will be used in real time in direct interaction with users. We hope to show that the Granska system with higher efficiency can achieve the same or better results than systems that use rule-based parsing alone.

1. Introduction

Grammar checking is one of the most widely used tools within language technology. Spelling, grammar and style checking for English has been an integrated part of common word processors for some years now. For smaller languages, such as Swedish, advanced tools have been lacking. Recently, however, a grammar checker for Swedish has been launched in Word 2000 and also as a stand-alone system called Grammatifix (Arppe 2000, this volume; Birn 2000, this volume).

There are many reasons for further research and development of grammar checking for Swedish. First, the need for writing aids has increased, both concerning the need for more efficiency and quality in writing. Secondly, the linguistic analysis in grammar checking needs further development, especially in dealing with special features in Swedish grammar and its grammatical deviations. This is a development that most NLP-systems will benefit from, since they often lack necessary methods for handling ungrammatical input. Thirdly,

there is need for more sophisticated methods for evaluating the functionality and usability of grammar checkers and their effect on writing and writing ability.

There are two research projects that focus on grammar checking for Swedish. These projects have resulted in two prototype systems: Scarrie (Sågvall-Hein 1998; Scarrie 2000) and Granska (Domeij, Eklundh, Knutsson, Larsson & Rex 1998). In this article we describe how the Granska system is constructed and how grammatical errors are handled by its error rule component. The focus will be on the treatment of agreement and split compound errors, two types of errors that frequently occur in Swedish texts.

2. The Granska system

Granska is a hybrid system that uses surface grammar rules to check grammatical constructions in Swedish. The system combines probabilistic and rule-based methods to achieve high efficiency and robustness. This is a necessary prerequisite for a grammar checker that runs in real time in direct interaction with users (e.g. Kukich 1992). Using special error rules, the system can detect a number of Swedish grammar problems and suggest corrections for them.

In figure 1 the modular structure of the system is presented. First, in the tokenizer, potential words and special characters are recognized as such. In the next step, a tagger is used to assign part of speech and inflectional form information to each word. The tagged text is then sent to the error rule component where error rules are matched with the text in order to search for specified grammatical problems. The error rule component also generates error corrections and instructional information about detected problems that are presented to the user in a graphical interface. Furthermore, the system contains a spelling detection and correction module which can handle Swedish compounds (Kann, Domeij, Hollman & Tillenius 1998). The spelling detection module can be used from the error rules for checking split compound errors.

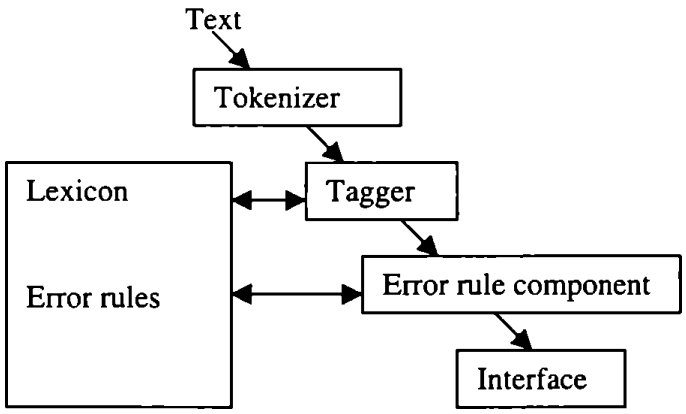


Figure 1. *An overview of the Granska system.*

The system is implemented in C++ under Unix and there is also a web site where it can be tested from a simple web interface (see www.nada.kth.se/theory/projects/granska/demo.html). There is ongoing work for designing a graphical interface for PC which can be used interactively during writing. The PC system will be used as a research tool for studying usability aspects with real users.

3. Tagging and lexicon

The Granska system uses a hidden Markov model (Carlberger & Kann 1999) to tag and disambiguate all words in the input text. Every word is given a tag that describes its part of speech and morphological features. The tagging is done on the basis of a lexicon with 160 000 word forms constructed from SUC, a hand tagged corpus of one million words (Ejerhed, Källgren, Wennstedt & Åström 1992). The lexicon has been further complemented with words from SAOL, the Swedish Academy's wordlist (Svenska akademien 1986). The Markov model is based on statistics from SUC about the occurrence of words and tags in context. From this information the tagger can choose the most probable tag for every word in the text if it is listed in the lexicon. Unknown words are tagged on the basis of probabilistic analysis of word endings.

4. Error rules

The error rule component uses special error rules to process the tagged text in search for grammatical errors. Since the Markov model also disambiguates and tags morphosyntactically deviant words with only one tag, there is normally no need for further disambiguation in the error rules in order to detect an error. An example of an agreement error is *ett röd bil* (a red car), where *en* (a) does not agree with *röd* (red) and *bil* (car) in gender. The strategy differs from most rule-based systems which often use a complete grammar in combination with relaxation techniques to detect morphosyntactical deviations (e.g. Sågvall-Hein 1998). An error rule in Granska that can detect the agreement error in ***ett röd bil*** is shown in rule 1 below.

Rule 1:

```
kong22@inkongruens
{
X(wordcl=dt),
Y(wordcl=jj)*,
Z(wordcl=nn & (gender!=X.gender | num!=X.num | spec!=X.spec))
-->
mark(X Y Z)
corr(X.get_form(gender:=Z.gender, num:=Z.num, spec:=Z.spec) Y Z)
info("Artikeln" X.text "stämmer inte överens med substantivet" Z.text)
action(granskning)
}
```

Rule 1 has two parts separated with an arrow. The first part contains a matching condition. The second part specifies the action that is triggered when the matching condition is fulfilled. In the example, the action is triggered when a determiner is found followed by a noun (optionally preceded by one or more attributes) that differs in gender, number or species from the determiner.

More formally, the condition part of the rule can be read as "an X with the word class determiner (i.e. wordcl=dt) followed by zero or more Y:s with the word class adjective (i.e. wordcl=jj*) and a Z with the word class noun (i.e. wordcl=nn) for which the values of gender, number or species are not agreeing with the corresponding values of the determiner X (i.e. gender!=X.gender | num!=X.num | spec!=X.spec). The characters "=","!=", "|" and "&" denotes the operators "is identical to", "is not identical to", "or" and "and" respectively. The comma is used for separating matching variables. The Kleene star (*) indicates that the preceding object can have zero or more instances.

Examples of phrases that match the condition is ***ett*** *röd bil* (deviation in gender), ***en*** *röda bilen* (deviation in species) and ***den*** *röda bilarna* (deviation in number).

The action part of the rule specifies in the first line after the arrow that the erroneous phrase X Y Z should be marked in the text. In the second line of the action part, a function (X.get_form) is used to generate a new inflection of the article X from the lexicon, one that agrees with the noun Z. When calling this function, the determiner X is assigned the same values of gender, number and species as the noun Z by the operator ":=" in order to get a new form from the lexicon that agrees with the noun. The new form is presented to the user as a correction suggestion (in the example ***en*** *röd bil*) by the corr statement. In the info statement in line 3, a diagnostic comment describing the error is constructed and presented to the user.

In most cases, the tagger succeeds in choosing the correct tag for the deviant word on probabilistic grounds (in the example *ett* is correctly analyzed as an indefinite, singular and neuter determiner by the tagger). However, since errors are statistically rare compared to grammatical constructions, the tagger can sometimes choose the wrong tag for a morpho-syntactically deviant form. In such cases, when the tagger is known to make mistakes, the error rules can be used in retagging the sentence to correct the tagging mistake. Thus, a combination of probabilistic and rule-based methods is used even during basic word disambiguation.

5. Help rules

It is possible to define phrase types like noun phrase (NP) and prepositional phrase (PP) in special help rules that can be used from any error rule. Rule 2 below, uses two help rules as subroutines (NP@ and PP@) in detecting agreement errors in predicative position. The help rules specify the internal structure of the NP and the PP in the main rule (pred2@predikativ). Note that the help rule PP@ uses the other help rule NP@ to define the prepositional phrase.

The main rule states that the copula X should be preceded by an NP optionally followed by zero or more PPs, and that an adjective Y that does not agree with the NP in gender or number should follow the copula. An example of a sentence matching the rule is *det lilla huset vid sjön är röd* (the little house by the lake is red) where the form *röd* does not agree in gender with the NP. The variables T and Z in the rule are contextual variables that

ensures that the NP is not part of a previous prepositional phrase and that the adjective Y in the supposed predicative position is not part of a larger noun phrase.

Rule 2:

```
pred2@predikativ
{
T(wordcl!=pp),
(NP),
(PP)*,
X(wordcl=vb & vbt=kop),
Y(wordcl=jj & (gender!=NP.gender | num!=NP.num)),
Z(wordcl!=jj & wordcl!=nn)
-->
mark(*)
corr(T NP PP X Y.get_form(gender:=NP.gender, num:=NP.num, spec:=ind) Z)
info("Substantivfrasen" NP.text "stämmer inte överens med
"adjektivet" Y.text)
action(granskning)
}


NP@
{
X(wordcl=dt)?,
Y(wordcl=jj*),
Z(wordcl=nn)
-->
action(hjälp, gender:=Z.gender, num:=Z.num, spec:=Z.spec, case:=Z.case)
}


PP@
{
X(wordcl=pp),
(NP)
-->
action(hjälp)
}
```

The help rules make the analysis approaches that of a phrase structure grammar. Help rules make it possible for the system to do a local phrase analysis selectively, without parsing other parts of the sentence that are not needed in the detection of the targeted error type. Thus, by calibrating the level of analysis that is needed for the case at hand the system obtains high efficiency.

6. Erroneously split compounds

Above we have shown how agreement errors are handled in the system. Another frequently occurring error type is erroneously split compounds. In contrast to English, a Swedish compound is regularly formed as one word so split compounds are treated as

ungrammatical. So far, we have mainly focussed on erroneously split compounds of the type noun+noun which stands for about 70 % of the various types (Domeij, Knutsson & Öhrman 1999).

Detection of erroneously split compounds where the first part cannot stand alone is trivial. This is done by listing those first parts in the lexicon and classifying them so that an error rule can be made to search the text for such a first part in combination with any other noun. An example is *pojk byxor* where *pojk* is the first part form of *pojke* (boy) which is combined with *byxor* (trousers).

In other cases when both parts have the form of full words, the strategy for detecting erroneously split compounds makes use of the fact that the first noun, unlike the last, must be uninflected (indefinite and singular). Since the combination uninflected noun followed by any noun is an unusual syntactical combination in grammatically correct sentences, it can be used to find candidates for split compound errors. Other contextual cues are also used before checking the candidate against a spell checker for compound recognition. If the spell checker recognizes the compound as such, the two nouns in the text are marked as a split compound and the corrected compound is given as a suggestion alternative.

Rule 3.

```
sär2@särskrivning
(
X1(wordcl=dt),
X2(wordcl=jj)*,
X3(wordcl=nn & (gender!=X1.gender | num!=X1.num | spec!=X1.spec)),
X4(wordcl=nn & gender=X1.gender & num=X1.num & spec=X1.spec &
correctly_spelled(concat(X3.text, X4.text)))
-->
mark(X3 X4)
corr(X1 X2 concat(X3.text, X4.text))
info("Särskrivningsfel, sätt ihop" X3 X4  "till " concat(X3.text, X4.text))
action(granskning)
)
```

In rule 3 above (which has been slightly simplified), an erroneously split compound is described where the determiner X1 does not agree with the first noun X3, but does agree with the second noun X4 as in the phrase *ett cykel ställ* (a bike rack). If the two nouns were to be combined into one word the result would be a perfectly grammatical phrase (*ett cykelställ*). Therefore, the disagreement between determiner and the following noun, together with the agreement between determiner and second noun give reasonable contextual evidence to suspect an erroneously split compound. To corroborate this hypothesis further, the two nouns are combined and checked by the spell checking function, as indicated in the last line of the condition, to see if the combined words are recognized as a compound. In the action part of rule 3, the error candidate is first marked in the text and then concatenated to be used in the error correction suggestion.

It can happen that two matching error rules collide, as in the example *ett cykel ställ* where both the rule for agreement error and the rule for erroneously split compound apply in an overlapping fashion. At the time, there is nothing to prevent the system to interpret

this error in both ways. However, the problem can sometimes be avoided by further disambiguation in the rules. For harder cases, a possibility would be to order the rules corresponding to the probability that an error occurs in a given context. Before we make a decision to implement such a function, we need to look deeper into the problem. Often, the wisest strategy is to present all error possibilities to the user.

Many errors can be difficult to detect because of ambiguities that are irresolvable on contextual grounds only. One example is *en exekverings enhet* (an execution unit). The first noun *exekvering* belongs to a group of nouns that take an –s when compounded with another noun (*exekvering-s+enhet*). When the compound is erroneously split, the form of the first noun coincides with the genitive form (an execution's unit) which has the same syntactical distribution as the error construction and therefore cannot be distinguished from the split compound case.

There are also problems with false alarms, for example when the tagger has mistagged a word so that the error rules make a false detection.

7. Results

The tagging module has a processing speed of more than 22 000 words per second on a SUN Sparc station Ultra 5. In a previously unseen text, 97 percent of the words are correctly tagged, a good result in an international comparison. Words not covered by the dictionary are correctly tagged in 92 percent of the cases. The whole system (with about 20 rule types for 250 error rules) processes about 2 800 words per second, tagging included. The numbers are hard to compare since results for other systems are seldom sufficiently presented, but we believe that we have achieved a comparably high performance. The results show that by using statistical methods it is possible to achieve a reasonably good linguistic analysis combined with high efficiency for real-time computation. We also believe that the analysis can be further improved by using rule-based methods for correcting faulty statistical analysis.

We are still working with optimizing the system and improving the error rules. Preliminary tests with the error rules show that we can hope for a recall rate above 50 percent and a precision rate above 90 percent for agreement errors and erroneously split compounds. The results so far are promising, but we need further development and testing before we present final results and compare them to fully developed systems, such as Grammatifix (Arppe, 2000, this volume; Birn 2000, this volume). Even if Granska is not yet fully developed, it has the advantage of being able to detect split compounds in Swedish, something that neither Grammatifix or any other commercial system does.

It is unrealistic to hope for full recall and precision. Therefore, we think it is important to test the program on users in practice to study usability aspects as well as the effects on writing and writing ability (see Domeij 1997, 1998). To do that we need to develop a user friendly and instructive graphical interface. The graphical interface for PC is scheduled to be ready during the spring 2000. The user tests will be ready before the end of the year.

Acknowledgements

The work has been funded by the Swedish research councils TFR, HSFR and Nutek. Project leader has been Kerstin Severinson-Eklundh. Språkdata at Göteborg University and the Swedish Academy let us use Svenska Akademiens ordlista as a source for words in Granska. Prof. Eva Ejerhed from Umeå University and Prof. Gunnel Källgren from Stockholm University let us use SUC.

References

Arppe, A. 2000. *Developing a Grammar Checker for Swedish*, Nodalida'99, Trondheim, December 1999.

Birn, J. 2000. *Detecting Grammar Errors with Lingsoft's Swedish Grammar Checker.* Nodalida'99, Trondheim, december 1999.

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

Domeij, R. 1997. Datorn och språkriktigheten. I: O. Josephson (ed.) *Svenskan och IT-samhället.* Uppsala: Hallgren & Fallgren.

Domeij, R., Eklundh, K., Knutsson, O., Larsson, S. & Rex, Å. (1998). Granskaprojektet 1996-1997. Technical Report NADA, KTH.

Domeij, R. 1998. Detecting, Diagnosing and Correcting Low-Level Problems when Editing with and without Computer Aids. In *TEXT Technology*, vol 8, no. 1. Wright State University, Celina, USA.

Domeij, R., Knutsson, O. & Öhrman, L. 1999. *Inkongruens och felaktigt särskrivna sammansättningar - en beskrivning av två feltyper och möjligheten att detektera felen automatiskt.* Svenskans beskrivning, October 1999.

Ejerhed, E., Källgren, G., Wennstedt, O. & Åström, M. 1992. The Linguistic Annotation System of the Stockholm-Umeå Corpus Project. Description and Guidelines. Version 4.31. Department of Linguistics, Umeå University.

Kann, V., Domeij, R., Hollman, J., & Tillenius, M. 1998. Implementaion Aspects and Applications of a Spelling Correction Algorithm. To appear in: R. Koehler, L. Uhlirova, G. Wimmer: *Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek*, 1999. NADA report TRITA-NA-9813, 1998.

Kukich, K. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, Vol. 24, No. 4, pp. 377-439.

Scarrie. 2000. Web site: stp.ling.uu.se/~ljo/scarrie-pub/scarrie.html

Svenska akademien (The Swedish Academy) 1986. Ordlista över det svenska språket (SAOL), 11th edition. Stockholm: Norstedts Förlag.

Sågvall-Hein, A. 1998. A Chart-Based Framework for Grammar Checking. *Proc. from Nodalida98.*

Adapting an English Information Extraction System to Swedish

Kristofer Franzén
Information and Language Engineering
Swedish Institute of Computer Science (SICS)
Box 1263, SE-164 29 Kista, Sweden
franzen@sics.se

This work was made possible thanks to generous funding from The Swedish Institute and The Swedish Foundation for International Cooperation in Research and Higher Education (STINT).

Abstract

This paper presents work on adapting the Proteus Information Extraction system to Swedish. It turned out that the cross-lingual adaptation as such was fairly straight-forward; however, the Proteus system design did not render itself that well to reconfiguration at such a low level as needed. To evaluate the adaptation, the system was tested on a Swedish version of the MUC-6 Scenario Template Task. The Swedish version performed excellently on a training corpus, but quite discouragingly on an unseen test corpus. As a consequence of that work, a new Information Extraction system is being designed and the layout of that system is described.

1 Introduction

A well-known problem in the area of Information Extraction regards the adaptation of an extraction system to handle a new class of events (Yangarber and Grishman, 1997). With the increasing interest in multi-lingual and cross-lingual information extraction, it becomes necessary to construct systems that are easily adaptable, not only to new extraction tasks, but also to new languages. This paper presents work on adapting the Proteus Information Extraction system (Grishman, 1995; Yangarber and Grishman, 1998) developed at New York University, to Swedish. The system has previously successfully been adapted to Japanese (Sekine and Nobata, 1998).

The topics covered in the following sections are: an introduction to the Information Extraction task, a description of the Proteus Information Extraction system, an account of the adaptations made to the system and some results from evaluating the adapted system. A description of our present work on designing a new information extraction system and the motivations behind it will conclude the paper.

2 Information Extraction

Information Extraction can be defined as the task of extracting instances of a predefined class of events from natural language texts, and to build a structured and unambiguous representation of the entities participating in these events and the relations between them.

While Information Retrieval (i.e., document retrieval) systems aim at returning a ranked list of documents as an answer to any arbitrary information need (posed in the form of a query), an Information Extraction system is tuned to a specific, well-specified, predefined and persistent information need. Input to the system is a stream of unrestricted text and it outputs a structured representation in the form of a filled template or database record for every instance of an answer to the information need.

In Figure 1, an actual information need that could be satisfied by an Information Extraction system is shown. The description is taken from the specification of the MUC-6 Scenario on Management Succession.

> "This scenario concerns events that would be of interest to an analyst who tracks changes in company management. The event object captures the management post, the company, the current manager, and the reason why the post is or will be vacant. The relational and low-level objects capture information on who's "in" and who's "out", where the new manager came from, and where the old manager is going. A relevant article refers to assuming or vacating a post in a company and must minimally identify the post and either the person assuming the post or the person vacating the post."

Figure 1: The narrative description of the MUC-6 Information Extraction Task.

A short text in Swedish and parts of the templates that could be filled in with information from the text, based on the above task definition, are shown in Figure 2.

Information Extraction and its methods of evaluation have to a great extent been defined by the Message Understanding Conferences (Grishman and Sundheim, 1996; Sundheim, 1991; Sundheim, 1992; Sundheim, 1993; Sundheim, 1995). The conference series is organized in the form of a competition where the participating extraction systems are evaluated against key templates constructed by human annotators. The metrics used to evaluate the systems are standard precision and recall measures over the template slots:

$$Precision = P = \frac{CorrectAnswers}{AnswersProduced} \qquad Recall = R = \frac{CorrectAnswers}{TotalPossibleCorrect}$$

These values are often combined into an F-measure:

$$F = \frac{(\beta^2 + 1)PR}{(\beta^2 P + R)}$$

Where β is a parameter that represents the relative importance of Precision (P) and Recall (R).

Karo Bio. Per-Olof Mårtensson har åter utsetts till VD efter att sedan förra våren ha varit ordförande. Mårtensson efterträds på ordförandeposten av Bertil Hållsten, tidigare chef för S-E-Bankens läkemedelsfonder.	**POSITION**	VD ('president')
	COMPANY	Karo Bio
	IN-PERSON	Per-Olof Mårtensson
	POSITION	ordförande ('chairman')
	COMPANY	Karo Bio
	IN-PERSON	Bertil Hållsten
	OUT-PERSON	Per-Olof Mårtensson
('Karo Bio. P-O M. has been reappointed president after serving as chairman of the board since last spring. Mårtensson is succeeded as chairman by B. H., former head of S-E-Banken's pharmaceutical funds').	**POSITION**	chef ('head')
	COMPANY	S-E-Bankens läkemedelsfonder
	OUT-PERSON	Bertil Hållsten

Figure 2: A short text and the three simplified templates it would generate.

As Appelt and Israel (1999) point out, interannotator agreement has been as low as 60-80% in the MUC:s (depending on MUC-task), which indicates that information extraction is a difficult task also for humans. They claim that, depending on the complexity of the extraction task and the preparation time, among other things, it seems very hard for an extraction system to reach beyond 60% of human accuracy.

Obviously, one of the main problems for an information extraction system is how to account for the linguistic variation in which the information is expressed in the text. This difficulty concerns lexical and syntactic variation as well as variation at the level of discourse and pragmatics. Consider the following constructed examples:

Assam Pärks nye VD Fjun Färneryd ...
(Assam Pärks' new CEO Fjun Färneryd ...)

Fjun Färneryd har utsetts till ny verkställande direktör för Assam Pärks AB.
(Fjun Färneryd has been appointed new chief executive officer of Assam Pärks AB.)

Fjun Färneryd, som igår utsågs till ny VD för Umeå-företaget Assam Pärks, ...
(Fjun Färneryd, who yesterday was appointed new CEO of the Umeå-based company Assam Pärks, ...)

F. Färneryd, 47 år och nybliven direktör för Assam Pärks, ...
(F. Färneryd, 47 years old and newly appointed president of Assam Pärks, ...)

Assam Pärks styrelse utsåg igår Fjun B. Färneryd till direktörsposten i ledningen för företaget.
(The board of Assam Pärks yesterday appointed Fjun B. Färneryd to the post as managing director of the company)

Assam Pärks har fått en ny VD. Fjun Färneryd satt tidigare i ledningen för Eckym Ropos, men fick lämna posten efter påståenden om insideraffärer.
(Assam Pärks has appointed a new CEO. Fjun Färneryd was earlier part of Eckym Ropos' management but had to resign after allegations of insider dealing.)

Just consider the difference in the first and the last example above, where, in the first case, a single noun phrase expresses relations that require supra-sentential inferential processing to deduct from the last example.

An Information Extraction system aims at text understanding, but only from the perspective relevant to the information need at hand. There is no need to resolve ambiguities in the text as long as they are not relevant to the present extraction task. Therefore most extraction systems make do with shallow parsing techniques (Grishman, 1995; Hobbs *et al.*, 1997) and local text analysis.

3 The Proteus Information Extraction System

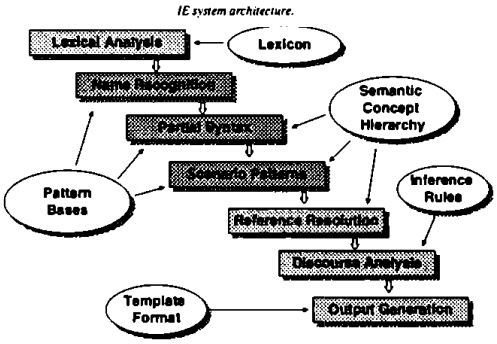


Figure 3: General architecture of the Proteus Information Extraction System.

New York University's Proteus system has a cascaded finite state transducer architecture common to many information extraction systems. It has a modular pipelined design consisting of domain-independent core components with domain-specific knowledge bases and task-dependent components for the specific scenario at hand. The lion's share of the linguistically interesting components of the system is defined in a number of pattern bases that are compiled at run-time into finite state transducers that perform deterministic, bottom-up, partial parsing and sequentially construct analyses of the text on increasingly higher levels of abstraction, building on the results from preceding transducers. The pattern bases contain rules that consist of a pattern part and an action part. A stylized rule to identify person names could look like

$$(Title)?\ CommonFirstName\ (MiddleInitial)?\ CapitalizedWord \rightarrow Tag\ as\ Person!$$

The rule would give *Mr. Fjun B. Ferneryd* an annotation of type Person which could be referred to in consecutive rules, e.g., in a pattern that identifies a company's appointment of someone:

$$NounGroup(type = Company)\ VerbGroup(type = Appoint)\ NounGroup(type = Person)$$

The general information flow through the system is shown in Figure 3 and the functions of the different modules are briefly described in Table 1 (see next page).

As can be seen in Table 1, some parts of an information extraction system are independent of the specific extraction task at hand, and some modules have to be modified when the system is tuned to a new task. This does not mean that the functionality of the component in itself should be changed, but rather that some rules or some knowledge contents that modify the behavior of the module have to be changed. The same should apply when porting an extraction system to a new language. For an English-Swedish bilingual system, there should be different rule sets for lexical analysis components, syntactic analysis components, and scenario specific patterns as well as for the pattern generalizations, but the rules guiding anaphora resolution and discourse analysis could possibly be the same; many of the knowledge bases could be shared, but the lexical leaves of the semantic concept hierarchy have to be re-mapped.

4 Changes made to the system

We aimed at changing the system as little as possible, but still get a reasonably good result. The Proteus system was adapted to the Swedish extraction task in the following ways

- *Input format.* Since the lexical analyzer and the tagger of the Proteus system were not to be used, an SGML interface to the system was constructed to facilitate the input from any external resources. For want of better alternatives, the SWECG tagger and disambiguator from Lingsoft (Karlsson *et al.*, 1995) was used for pre-processing the Swedish text, which then had to be postprocessed to deal with the inconsistent SWECG output. The output was then transformed into the SGML format.
- *Rule predicates.* A rule in the system consists of a pattern matching part and an action part. Some of the predicates used in the pattern matching part of the rules where modified to allow for richer descriptions of the matched elements. Minor adjustments had to be made for Swedish (Latin-1) characters to be accepted in the pattern matching rules and their actions.
- *Domain and task independent rules.* Patterns for noun groups and verb groups had to be redefined, as well as patterns to identify, for example, people, organizations and locations.

Module	Description	Scenario Specific
Core modules		
Lexical analysis	Assigns part-of-speech tags to the text	no
Name detection and categorization	Identifies person names, company names, names of locations and possibly products	no ? no ?
Analysis of numerical expressions	Identifies monetary expressions, percentages and time/dates	no
Noun group detection	Identifies noun phrases without right modifiers.	no
Verb group detection	Verb + auxiliaries to identify main verb and tense.	no
Noun phrase detection	Full noun phrases for important scenario entities.	yes
Anaphora resolution	Resolution of pronouns and definite nouns involved in the scenario.	no
Scenario specific pattern matching	Top-level patterns for the specific extraction task.	yes
Discourse analysis	Co-reference analysis on the discourse level to merge events.	?
Inference rules	Formalizes world knowledge in rules so that text content fits template format. I.e., "last week" becomes a date, etc.	yes
Template generation	Produces the filled template for the specific task.	yes
Supporting modules		
Knowledge bases	Lists of common first names, corporations, locations and scenario specific entities.	no yes
Semantic Concept hierarchy	A hierarchy of concepts to support pattern matching, anaphora resolution and discourse analysis.	yes
Pattern production module	Allows for the user to produce scenario specific patterns interactively from examples.	no
Pattern generalization module	Meta-patterns that generalize patterns to match various kinds of subjunctive clauses, reduced clauses, passives etc.	no

Table 1: Common modules of an Information Extraction System.

- *Task specific rules.* Patterns for *entities* participating in the *events* of the task had to be redesigned, as well as the patterns for the *events* themselves.

- *Knowledge bases.* Several knowledge bases specific to Swedish were compiled to support the identification of names of people, organizations, locations and reportable positions,[1] etc.

The difficulties in trying to adapt the Proteus system to Swedish were not the linguistic differences (as expressed, for example, in the shallow parsing pattern matching rules), nor the differences in how the events were expressed in the different languages (as is expressed in the higher level rules); these patterns and rules were often surprisingly interchangeable, with small modifications, across the languages. What posed severe problems were the technical difficulties in changing a very complex system that was not initially built to be reconfigured on such a low level. Even though there is a graphical user interface to the English system in which the user can build patterns incrementally from examples, that tool would have required extensive work to function with the SGML input format and the other modifications made to the system.

5 Experiment and evaluation

For a comparison of the performance of the Swedish and the English systems, the Scenario Template Task of MUC-6 was chosen. This task concerns changes in corporate executive management personnel, as described in Figure 1. A Swedish corpus was compiled consisting of 34 financial news articles from *Tidningarnas Telegrambyrå* and *Affärsvärlden*. This training corpus contained 51 reportable events for which key templates were constructed. Rules were written and evaluated iteratively with the MUC-scorer[2] on the corpus until an F-score of 55.45 was obtained. In comparison, the systems at the MUC-6 evaluation ranged from about 48 to 56 in F-score on the test corpus (Sundheim, 1995). After extensive training, the Proteus system has since been boosted to perform at an F-score around 65 on the same task.

A test corpus consisting of 50 financial news articles from the same sources as the training corpus was compiled, as well as template keys, by an annotator not involved in the adaptation of the extraction system. The results from running the system on the test corpus seem quite discouraging with an F-score of around 28, but have not yet been fully analyzed. Further analysis will show if they are due to over-training of the system, faulty system design, or mismatches in the annotator's and the author's interpretation of the template filling rules.

6 Building a new system

The overall experience of trying to adapt the Proteus Information Extraction system to Swedish has led to the decision to build a new Information Extraction system. This

[1] For example, according to the MUC-6 definitions, 'chairman of the board' is a reportable position while other types of chairpersons are not reported.

[2] The MUC-scorer is described in http://www.muc.saic.com/scorer/Manual/manual.html

system will be inspired by the general architecture of the Proteus system, but also by suggestions of improvements of that system that came up during and after the work cited in this paper. The new system will be built around a document manager which functionality is a subset of that in the Tipster Architecture (Grishman and others, 1996). This means that all internal functions are based on manipulating annotations of the text. We will aim at giving the system the following features:

- *Easily portable to new domains.* We recognize the need for a tool that facilitates for the non-expert user to write rules for a new extraction task without knowing the internals of the system or the syntax of the pattern matching language. Such a tool for example-based pattern acquisition exists in the Proteus system (Yangarber and Grishman, 1997).

- *Easily portable to new languages.* We will make every effort not to build language prerequisites into the system. For example, there will not be any restrictions on what features or feature values that may be found in an annotation.

- *Easily extensible.* Since there will be a well-defined interface to the document manager and a general set of methods to manipulate document annotations, and since there will be no restrictions on what the features of the annotations can be, the system is not limited to merely Information Extraction tasks, but can be extended to any document or text manipulating task.

- *Modular and flexible.* The system will have an object oriented design with distinct interfaces between the modules. If a new module is required for an analysis task, it should be easy to include it in the existing set of modules.

- *Platform independent.* The system will be implemented in Java and not dependent on external software in itself. Initially, the extraction system will be dependent on an external lexical analysis component.

- *Open Source.* Every part of the core extraction system will be open source and free to use for research or commercial purposes.

7 Conclusion and Further Work

Even though the Proteus system has previously successfully been adapted to Japanese, our experiences in adapting the system to Swedish have led us to believe that it will be worth the effort to build a new extraction system from scratch, taking into account portability not only on the task level, but also on the language level. Such a project has been initiated in collaboration with New York University and the result will eventually be publicly available.

This work has also led to the composition of the above mentioned test corpus for Swedish Information Extraction systems which will also be publicly available as soon as copyright issues are solved.

References

Appelt, D. E. and Israel, D. J. 1999. Introduction to Information Extraction Technology. http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf. A Tutorial Prepared for IJCAI-99.

Grishman, R. 1995. The NYU system for MUC-6, or where's the syntax? In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November. Morgan Kaufman.

Grishman, R. and Sundheim, B. 1996. Message Understanding Conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING96)*, Copenhagen, August.

Yangarber, R. and Grishman, R. 1997. Customization of Information Extraction Systems. In *Proceedings of International Workshop on Lexically Driven Information Extraction*, Frascati, Italy, July.

Yangarber, R. and Grishman, R. 1998. NYU: Description of the Proteus/PET System as Used for MUC-7 ST. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Virginia, USA, April. Morgan Kaufman.

Hobbs, J. R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., and Tyson, M. 1997. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In Roche, E. and Schabes, Y., editors, *Finite-State Language Processing*, Language, speech, and communication. MIT Press, Cambridge, Massachusetts.

Karlsson, F., Voutilainen, A., Heikkilä, J., and Anttila, A., editors. 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text.* Mouton de Gruyter, Berlin.

Sundheim, B., editor. 1991. *Proceedings of the Third Message Understanding Conference (MUC-3)*. Morgan Kaufman, May.

Sundheim, B., editor. 1992. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufman, June.

Sundheim, B., editor. 1993. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*., Baltimore, MD, August. Morgan Kaufman.

Sundheim, B., editor. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November. Morgan Kaufman.

Sekine, S. and Nobata, C. 1998. An Information Extraction System and Customization Tool. In *Proceedings of the New Challenges in Natural Language Processing and its Application*, Tokyo, Japan, May 25-26.

Grishman, R. et al. 1996. TIPSTER Text Phase II Architecture Design. Technical report, Department of Computer Science, New York University, September.

The shortcomings of a tagger

Kristin Hagen, Janne Bondi Johannessen og Anders Nøklestad
The Text Laboratory, The University of Oslo
{kristiha, jannebj, noklesta}@hedda.uio.no

Abstract

The tagger used for the Oslo Corpus of Tagged Norwegian Texts has very good statistical results. In spite of this, it makes mistakes. In this paper we take a closer look at some of them. Although some mistakes are of a kind that would disappear if we improved the tagger, many are impossible or very difficult to do anything about. They are due to errors in the corpus (spelling errors, foreign words, non-standard spellings), to elliptic sentences, such as headlines, and to structural ambiguity, which abounds to a surprising extent. Proofreading the corpus would have removed the first kind of problems, but the other two types cannot be resolved in any obvious way.

1. Introduction

The first version of the first ever comprehensive tagger for Norwegian is ready. Both the *nynorsk* and the *bokmål* (the two Norwegian language varieties) versions have been used to tag a large number of texts (= the Oslo Corpus of Tagged Norwegian Texts). The corpus has an advanced web-based user interface, which often gives nice results, but it also makes it easy to discover mistakes and shortcomings of the tagger. The present paper will focus on these.

The tagger is of a Constraint Grammar-type (Karlsson et al 1995). The linguistic constraints (rules) were developed by the Text Laboratory, while the software came from Lingsoft, Helsinki. A CG tagger takes as input a multitagged text, where each word form has as many tags as the lexicon allows, and gives as output a text where the tags are disambiguated by the given linguistic constraints according to the context for each word in question. The statistical results are good: The bokmål tagger has a recall of 99,2% and a precision of 96,8%. For nynorsk the results are slightly worse: 98,8 % recall and 95,6 % precision.

The tagger, then, makes some mistakes. One kind of shortcoming involves cases where some ambiguity remains (this influences the precision rate) - for a number of reasons, of which structural ambiguity is the most severe one: Sometimes extralinguistic knowledge would be required to disambiguate a certain ambiguity. Another shortcoming has to do with mistaken lexical analysis: We have problems when a text contains words that are unknown to the lexicon or that are analyzed wrongly by our compound analyzer, or if they even contain a wrong language (common in citations, loanwords etc.).

Before we go into these mistakes, however, let us give an example which shows that in spite of the errors, the overall impression is that the tagger actually does a good job. In the following example, we have asked the corpus to give us all occurrences of the word *stemme* ('vote') used as a verb; we therefore do not want any occurrences of the same wordform used as a noun meaning 'vote' or 'voice'. And indeed, the overall impression is that we get what we wanted:

(1) Example of an arbitrary selection of hits from a search for the verb *stemme* 'vote', as opposed to the noun *stemme* 'vote' or 'voice'

Søkestreng: [word="stemme" & tagg=".* verb.*" & (src="AV.*" | src="SA.*" | src="SK.*")] med 30 tegn på
venstre side og 40 tegn på høyre side.

AV/Ad96/01: re hele det norske folk ved å *stemme* mot, eller utfordre NATO-kolleger ved
AV/Ad96/01: utfordre NATO-kolleger ved å *stemme* for. I slike saker gjelder diplomatiet
AV/Ad96/01: re hele det norske folk ved å *stemme* mot, eller utfordre NATO-kolleger ved
AV/Ad96/01: utfordre NATO-kolleger ved å *stemme* for. I slike saker gjelder diplomatiet
AV/Ad96/01: møter opp, men avstår fra å *stemme*, vil i praksis støtte eventuelle fusjo
AV/Ad96/01: og Mosvold Farsund Invest vil *stemme* for fusjonen. - Gjensidige sier ja til
AV/Ad96/01: n med Danmark og Island fra å *stemme* da FNs nedrustningskomite i går
AV/Ad96/01: var to var i tvil, én ville *stemme* nei, men bademester Eivind Nilsen (ek
AV/Ad96/01: re, sa begge at de kom til å *stemme* for svensk medlemsskap i EU. Eivind
AV/Ad96/01: e, mens ingen tidligere fikk *stemme* for mer enn 12,5 prosent. I forbindels
AV/Ad96/01: egjering som et mottiltak vil *stemme* nei til alle reformer av EU-samarbeidet
AV/Ad96/01: ens 38 land avholdt seg fra å *stemme*. Resolusjonen er ikke bindende, men
AV/Ad96/01: neringen, men fikk det til å *stemme* rimelig bra mot TPG, sa Håvard. - Vi
AV/Ad96/01: var to var i tvil, én ville *stemme* nei, men bademester Eivind Nilsen (ek
AV/Ad96/01: re, sa begge at de kom til å *stemme* for svensk medlemsskap i EU. Eivind
AV/Ad96/01: spennende hvordan dette ville *stemme* med søkernes ønsker. Vi så for oss
AV/Ad96/01: sjonelle ferdighetene » til å *stemme* igjen. Han har 14 dager på seg før den
AV/Ad96/01: tradisjonelt hatt for vane å *stemme* sammen med Arbeiderpartiet og SV.
AV/Ad96/01: ndre tusen nye velger som vil *stemme* på meg! Sjakk i UKE-Adressa | En
AV/Ad96/01: dene hadde forsiktig begynt å *stemme* sine bakben (som de gnir mot en
AV/Ad96/01: den britiske regjering med å *stemme* nei i EUs ministerråd i alle spørsmål,
AV/Af94/01: ene søkeren at hun kom til å *stemme* på ham dersom hun deltok på møtet,
AV/Af94/01: peacemedlemmer fullmakt til å *stemme* på landets vegne. Selv var de ikke til
AV/Af94/01: pålegge sine representanter å *stemme* efter en vedtatt partilinje. Men konse
AV/Af94/01: nde. Hvordan skulle da Høyre *stemme* i Stortinget? Partilandsmøtet løste sa
AV/Af94/01: ervere seg dersom de ikke kan *stemme* for i Stortinget, sier hun. Efter man
AV/Af94/01: t partiet Rukh sier at de vil *stemme* imot avtalen når den kommer opp i parla
AV/Af94/01: år. På Lillehammer skal alt *stemme* i første forsøk, legger han til. - Ko
AV/Af94/01: en i fjor, sier nå at de vil *stemme* Høyre igjen. Høyrelederens opptreden s
AV/Af94/01: ripe inn i tidens politikk og *stemme* med Venstre i unionsstriden også når ha
AV/Af94/01: r frem til velgerne går for å *stemme*. Og kommer velgere som vil ha en

2. Structural ambiguity

Each time a word is left ambiguous between two categories, the corpus user will tend to think that the tagger is unsatisfactory. However, there is a lot of structural ambiguity in language. Most of it goes unnoticed, because our pragmatic and world knowledge guides us towards the right interpretation. But a tagger has only access to form, i.e. morphology and syntax, and will not be able to know which interpretation is the correct one when the formal features are the same.

Let us look at some examples.

(2)

Norwegian, BM:	Jeg kjente meg glad og *lettet* da hun gikk
Two readings:	a. I felt happy and *relieved* when she left
	b. I felt happy and *took off in the air* when she left
Ambiguity:	*lettet: "lette" 'take off in the air' verb pret*
	"lette" 'relieved' adj masc fem ind sg

Our world knowledge tells us that the pronoun *jeg* refers to a person, and we also know that people do not have wings, and therefore normally will stay on the ground, unless something in the context tells us otherwise. We also know that the feeling of being *glad* ('happy') often goes together with the feeling of being *lettet* ('relieved'). As human beings, we therefore interpret the sentence in (2) in the only pragmatically correct way; the a-reading. But the tagger has no world knowledge, and must leave the sentence ambiguous, i.e. leave the word *lettet* with both tags.

(3)

Norwegian, NN:	Vaskehjelper som vaskar *skular...*	
Two readings:	a. Cleaning women/men who wash *schools...*	
	b. Cleaning women/men who wash *stare...*	
Ambiguity:	*skular:*	*"skule" 'schools' noun plural indef*
		"skule" 'stare verb present tense

When seeing a sentence like (3), we know immediately that the relative clause would be a tautology if it only contained the verb without its object. We therefore understand the last word as the object of the verb rather than as a verb. But the tagger is in no position to decide which of the meanings would be meaningless, and has to leave the word *skular* with both tags.[1]

(4)

Norwegian, NN:	Ho skulle sleppa fara på åker og eng, *berre* ho ville sjå til huset	
Two readings:	a. She would not have to travel in fields and meadows, *if only* she would look after the house	
	b. She would not have to travel in fields and meadows, she was the only one to look after the house	
Ambiguity:	*berre:*	*"berre" 'if only' subjunction*
		"berre" 'only' adverb

In (4), we understand that the most likely interpretation is that the second clause is a condition for the first clause. But the tagger finds the second reading, in which the second clause is a juxtaposed main clause, just as likely. Therefore, the word *berre* must be left ambiguous, keeping both the subjunction and the adverb tags.

(5)

Norwegian, NN:	Ho kysste han gang på gang før ho og vart riven bort
Two readings:	a. She kissed him time and time again before her and was taken away
	b. She kissed him time and time again before she too was taken away

Ambiguity:	*før:*	*"før" 'before' preposition* *"før" 'until' subjunction*
	og:	*"og" 'and' conjunction* *"og" 'too' adverb*
	ho:	*"ho" 'she' pronoun nominative* *"ho" 'her' pronoun accusative*

In (5), both readings are equally likely without knowing more about the context. In the a-reading, there are two women involved, where one kissed the male before the other one, and was subsequently taken away. In the b-reading, there is only one woman, who kissed the man until she - in addition to somebody else - was taken away. There is no way the tagger would be able to choose betwen these readings, and three words have to be left ambiguous as a result.

(6)

Norwegian, BM:	Smidsrød har *arbeidet som forsker* ved NTNFs Norsk Institutt for Tang- og tareforskning fra 1961	
Three readings:	a. Smidsrød has worked as researcher at NTNF... b. ??Smidsrød has (his) work which does research... c. ?? Smidsrød has (his) work as researcher...	
Ambiguity:	*arbeidet:*	*"arbeide" 'work' verb past participle* *"arbeid" 'work' noun sg def*
	som:	*"som" 'as' preposition* *"som" 'which' relative subjunction*
	forsker:	*"forsker" 'researcher' noun sg ind* *"forsker" 'research' verb pres*

In (6), the meanings of the italicized words tell us that the word *arbeidet* should be interpreted as a verb. E.g., we know that *arbeidet* can never be an agentive noun, and therefore never be the subject of a verb *forsker*. We also know that the italicized words should not be interpreted as a noun phrase, as would have been the case in e.g. *Smidsrød har arbeidet som hobby*. Again, the tagger cannot choose, and will have to leave three words ambiguous.

3. Headlines have too little grammatical information

Headlines and titles generally are very rudimentary sentences that often lack a verb and function words. There is therefore very little information that can guide the tagger in the right direction when it comes to choosing between different readings:

(7)

Norwegian, BM:	Rushfeldt *for dyr* for Viking?	
Two readings:	a. Rushfeldt *too expensive* for Viking b. Rushfeldt *for animals* for Viking	
Ambiguity:	*for:*	*"for" 'for' preposition* *"for" 'too' adverb*

dyr: *"dyr" 'animals' noun ind plural*
"dyr" 'expensive' adjective ind sg masc

As Norwegians, we know that Rushfeldt is a footballplayer, that Viking is a football club, and that football players often require a lot of money to change clubs. The a-reading is the only appropriate one. But the tagger does not know that the other reading is impossible, in which Rushfeldt would make a statement in favour of animals. Two tags are left ambiguous as a result.

(8)

Norwegian, BM: Luftpistol - Internasjonal *gren* og olympisk øvelse
Two readings: a. Air pistol - International branch and olympic event
b. *Air pistol - International cried and olympic event

Ambiguity: *gren:* *"grine" 'cry' verb preterite*
"gren" 'branch' noun ind singular masc

We know that, although an adjective can be the subject of a clause in Norwegian, in this particular sentence, which has to to with Olympic events, it is obvious that the word *gren* refers to air pistols a a branch, and is not a verb. But since this is a headline, there is no requirement for a finite verb, and indeed it does not have one, which might otherwise have helped disambiguate this word. And with no world knowledge, the tagger cannot choose between the two readings.

4. Wrong language or dialect causes problems

In an open text corpus there will always be examples of words and phrases that belong to other languages and dialects. We have not wanted to clean the corpus of this type of occurrences. Obviously, then, words from other languages will not be correctly analyzed by our monolingual tagger. This may in turn create problems for the tagging of the other words surrounding the unknown word - since disambiguation to a large extent depends on the local context of each word. Below are some examples of foreign elements:

(9) Dialect: Uknown word causes unresolved ambiguity in preceding word:

Norwegian, BM: Det *va* et godt forslag. Deinn første kjæresten m...
(*va* instead of *var*)
Two readings: •It was a good proposal
•The was a good proposal

Ambiguity: *det:* *"det" 'the' determiner demonstrative sg neuter*
"det" 'it' pronoun sg neuter

In (9), the italicized word *va* is a Trøndelag dialect word for the standard word *var* (preterite of the verb *være* 'be'). Since *va* is the infinitive form of a verb meaning to walk in water, the tagger finds no finite verb. It will then not know that the first word is a subject, and will not be able to understand that it is the pronoun reading, and not the determiner one, that should be chosen for *Det*. It is left ambiguous.

Other languages and dialects are actually quite common in texts generally - here are some more examples:

(10) German word:

Det er *«schönt»*, for han var mitt forbilde, sier Anders.
It is *wonderful* for he was my idol, Anders says.

(11) Trondheim dialect word:

Ni straffekast til Old Girls og bare tre til *ungpian* startet kampen.
Nine penalties for the Old Girls and only three to the young lasses started the match.

(12) Nynorsk Norwegian sentence in a Bokmål text:

Zimmer-utvalet har kome med framlegg til ny lov her til lands om eigedomsskatt.
The Zimmer committee has suggested a new law in this country about property tax

5. Words written in a way that is believed to be right, but isn't

Orthography is not easy, and indeed lots of people are unaware of how to write or even how to inflect certain words according to the norm. The tagger uses lexicons that follow the standard norm (Bokmålsordboka, Nynorskordboka, IBM's lexical database). Although we have made an effort to enlarge our lexical database to include the most common misconceptions (see Hagen, Johannessen and Kristoffersen 1997), it is not possible to foresee all possible mistakes, as can be seen below.

(13) A phrase believed to be a compound

Norwegian, NN: Det er ikkje nokon *kvensomhelst* som no står fram som ja-mann
Wrong analysis: • *It is not any *who-who-rather* who now stands forward as a yes-man
(should have been *anybody*)

Wrong analysis: compound adverb	*kvensomhelst:*	*"kven-som-helst" 'who-who-rather'*
Should have been:	*kven som helst*	*"kven som helst" 'anybody'*

The way Norwegian creates 'free choice items', like the English *any*, is by adding the (untranslatable) phrase *som helst* to the word in question. Since this is a set phrase, it may easily be conceived of as being compounded with the word it modifies. This has happened in this particular context. Since the word is not in the lexicon, the compound analyzer belonging to the tagger has, correctly, found the three words it consists of, but has treated it like all other compounds, giving the compound as a whole the tag of its last member. This of course gives the wrong result: The compound is given the tag adverb rather than pronoun, or even noun in this particular context. (Indeed, this particular word probably ought to have been added to the lexicon as a nominal compound.)

(14) Wrong inflection

Norwegian, NN:	Ho er fødd i Kristiansand og *vaks* opp på Gjøvik
Wrong analysis:	She was born in K. and *of the fish being on the feed* up at Gjøvik. (should have been: *voks*)

Wrong analysis:	*vaks:*	*"vak" 'fish being on the feed near the surface' noun sg gen*
Should have been:	*voks:*	*"vokse" 'grow' verb preterite*

Strong verbs - verbs that inflect with an internal vowel change - sometimes have an inflectional norm that does not comply with what people actually believe to be the case. In (14) the verb is therefore analyzed by the tagger as a noun, which of course will prevent the correct analysis and disambiguation of the rest of the sentence.

These mistaken beliefs are actually very common. Actually, the Oslo Corpus contains 96 occurrences of the wordform *vaks* (supposed to be preterite of the verb meaning "grow"), compared to 145 correct ones. In other words, 40 per cent of the occurrences are written in a nonstandard way. The same is true in the results from a general Alta Vista web search: 173 pages contain the form "vaks " and 840 the form "voks ". Given that many of the latter ones also must have belonged to the homonymous noun meaning "wax", we can conclude that this mistaken belief with regard to spelling is very common.[2]

6. Spelling errors and mistakes generally cause problems

Every time there is a mis-spelt word or mistakes in punctuation, the tagger will have problems. A mis-spelt word will either not be analyzed or be analyzed wrongly, with the result that other words surrounding that word will also be difficult to analyze. For example, if a noun is wrongly identified as a verb, then the determiner of that noun will not be analyzed correctly, since a determiner needs a noun to be identified. If there is a mistake in punctuation, the tagger will not know where the clause ends. This has serious consequences. Since the tagger, for example, accepts only one finite verb for each clause, a missing full stop will make it impossible to identify two finite verbs in what is really two clauses.

(15) Lack of full stop:

Norwegian, BM:	Du kan også svare på fax : 72501468 eller via e-mail : ole-einar.andersen@adresseavisen.no *Vi* må ha svaret innen kl. 12.00.
One wrong reading:	You can also answer by fax:... or by e-mail:... Vi must have the answer by 12 Midday.

Wrong analysis:	*Vi:*	*"Vi" proper name*
Should have been:	*Vi:*	*"vi" 'we' pronoun pl*

In (15), the word *Vi* is of course a pronoun that is written with a capital letter because it is the first word of a sentence. Our knowledge of language makes it immediately possible to interpret it correctly, and to spot that there is a missing full stop in front of this word. However, the tagger has more limited knowledge, and instead analyzes this unknown word as a proper name, wrongly of course, with bad results for further identification of the words in the clause.

Below are a couple of more examples of printing and spelling errors that are problematic:

(16) Two words written together:

I Trondheim er *mellom30* og 50 stellebord av denne typen solgt.
In Trondheim, *between30* and 50 changing units of this kind have been sold.

(17) Wrong spelling:

Hvis betingelsene for *forskninng* er bedre i andre land enn her hjemme, vil forskningen etter hvert flyttes ut. (should have been: *forskning*)

If the conditions for research are better in other countries than here at home, research will be moved out after a while.

7. Wrong for other reasons

There are cases in which the tagger would have had better results had we improved it in certain ways. Below are a couple of such examples.

(18) A word unknown to the lexicon

Norwegian, BM:	... i Europa. *Per-Åke* Palmquist som alle de andre...
Two readings:	• *...in Europe. *Per-go* Palmquist like all the others...
	• ...in Europe. *Per-Åke* Palmquist like all the others...
Ambiguity:	*Per-Åke: proper name*
	Per-Åke: "per-åke" 'per-go' verb infinitive

If a word with a capital letter follows a full stop, it is possible to analyze it as a proper name if the word is not in the lexicon. But if the word is ambiguous between a proper name and a word in the lexicon, or is a possible compound, it is more difficult for the tagger to make the right choice. In (18), the name is interpreted as a compound, since the last part of it could be a verb. A list of names or a statistical module telling the tagger that the verb *åke* is very rare might have solved this problem, but as it stands, without these, the problem remains.

(19)

Norwegian, NN:	Han kjende berre *noko voks* oppunder ermstaupet på han.
Only reading:	a. He only felt *some wax* under his armpit.
Not analyzed:	b. He only felt something grew under his armpit
Ambiguity:	noko: "noko" 'something' pronoun neuter sg
	"noko" 'some' determiner neuter sg
	voks: "vokse" 'grow" verb preterite
	"voks" 'wax' noun sg ind masc

In (19), the problem for the tagger is that it has to understand that the italicized words, in addition to being a noun phrase consisting of a determiner plus a noun, can also be analyzed as a pronoun followed by a relative clause without a relative subjunction. But

this second reading is overall less likely, and to accept this kind of reading would probably give many more ambiguities in the rest of the tagging.

8. Conclusion

As long as each text is not cleaned before tagging, some problems are bound to remain unsolved. We have chosen this inclusive perspective for The Oslo Corpus because we believe that our users appreciate the possibility of being able to do searches in a large corpus. If we had chosen a restrictive attitude to the way the corpus texts should look before they were taggable, our corpus would have been considerably smaller, because we would have had to proofread it. The tagging mistakes which are due to wrong spelling and to wrong language and dialect are therefore impossible to prevent.

Some of the mistakes are due to people's mistaken beliefs. This kind of mistake, which is finite in number, can be accommodated by expanding the lexicon to include nonstandard spellings and inflections of words. We have already done this to some extent, and we have also done the opposite - reduced the lexicon by removing some extremely infrequent correct wordforms that are homonymous with some very frequent ones.

Structural ambiguity and ambiguity due to headlines are two problems that we do not see that we can solve. They require world knowledge of a kind that is hard to include in even very domain specific AI systems, and are impossible to include in tagging of completely open text corpora.

The fact that there turns out to be a surprising amount of structural ambiguity, however, is interesting with respect to the evaluation of taggers more generally. There are basically two types of taggers: those that leave ambiguity where it cannot be decided, like the Constraint Grammar type (Karlsson et al 1995) that we have used for the Oslo Corpus, and those that always make a choice, like statistical taggers (e.g. Kupiec 1992). It is possible that languages differ with respect to how much structural ambiguity they allow. We believe, after having worked with tagging of Norwegian, that a tagger which allows structural ambiguity to remain unsolved is preferable to one that does not.

Notes

1. One might ask whether the rest of this sentence would make the exerpt unambiguous, but this is not the case:

(i) Reinhaldsarbeidarar eller vaskehjelper som vaskar *skular*, kommunehus, bibliotek og andre kommunale hus bør få nøye opplæring i korleis dei skal utføre arbeidet sitt.

'Cleaning personell or washing people who wash [a. schools/b. stare], city halls, libraries or other buildings belonging to the council ought to be taught how to perform their work properly.'

With the a-interpretation, the noun *skular* is in a multiple coordination with other kinds of buildings, all being part of the object of *vaskar*. With the b-interpretation, the verb *skular* ends the clause, while the other kinds of buildings are the (pragmatically odd) subject of a second, asyndetically coordinated, main clause.

2. We are grateful to Øystein Alexander Vangsnes for making us aware of these facts.

References

Bokmålsordboka. 1993. Landrø, M.I and B.Wangensteen (ed.). Universitetsforlaget, Oslo.

Hagen, K., J.B. Johannessen and K.E. Kristoffersen. 1997. Problemer ved bruk av andres lister til taggerformål. Paper presented at Møter om norsk språk 7, University of Trondheim.

Karlsson, F., A. Voutilainen, J. Heikkilä and A. Anttila. 1995. *Constraint Grammar.* A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin.

Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model, *Computer Speech and Language* 6, 225-242.

Nynorskordboka. 1998. Hovdenak, M., L. Killingbergtrø, A. Lauvhjell, S. Nordlie, M. Rommetveit and D. Worren (red.), Samlaget, Oslo.

The Oslo Corpus of Tagged Norwegian Texts:
http://www.tekstlab.uio.no/norsk/bokmaal/
http://www.tekstlab.uio.no/norsk/nynorsk/

Merging Classifiers for Improved Information Retrieval

Anette Hulth, Lars Asker
Dept. of Computer and Systems Sciences
Stockholm University
[hulth|asker]@dsv.su.se

Jussi Karlgren
Swedish Institute of Computer Science
jussi@sics.se

Abstract

One prospective way to improve information retrieval is to use several indexing methods to retrieve different sets of documents, and then to merge (or combine) these results into one single result. The merging should be done in a way that produces a final result that is more accurate than the output of any of the individual classifiers. A merging algorithm called *SEQUEL* has been applied for this task to data in the field of information retrieval. This article describes the results of these experiments, as well as conceivable future directions.

1 Introduction

Training several different classifiers and combining their predictions into a single one is a common method for creating a classifier from a set of training data. This approach generally yields a more accurate result than that from the constituent classifiers, which has been shown by a number of researchers. (For an overview of research and results in this area see for example Merz (1999) and Dietterich (1997).) Similar results have also been shown in the document retrieval domain by Bartell *et al.* (1994): using different retrieval algorithms and then combining them may significantly improve retrieval performance.

When applying a merging strategy, the first thing to decide is what different classifiers to use. In the information retrieval domain the source of information is often documents. As documents consist of words, a feasible approach could be to use linguistic methods for retrieval. If each of the methods captures a different aspect of the documents' content, we could possibly retrieve a larger amount of relevant documents on the whole. When merging the different results, the aim is to produce a final result that is more accurate, i.e., has a higher average precision, than the output of any of the individual results. Obtaining this is, however, not trivial, as we only have recourse to weak clues about text relevance, and since results vary between queries, domains and reader preferences, all of which is based on knowledge that is difficult to model reliably.

2 Background

In this section we will describe the Text REtrieval Conference—the framework within which most of the work was done. We will also describe how the relevance is judged, as well as give a motivation for using several information retrieval methods.

2.1 Text REtrieval Conference

The Text REtrieval Conference (TREC) is an annual workshop on information retrieval organised in the form of a competition by National Institute of Standards and Technology (NIST). Several aspects of information retrieval are dealt with within different tracks: *interactive search* (using users' feedback); *spoken document retrieval*; *question answering* (the answer to a stated question is returned to the user); *cross-language retrieval* (the database is multi-lingual), just to name a few. The track that attracted the largest number of participants at TREC-8 was the *ad hoc* track. This task investigates the performance of systems that search a static set of documents using new topics. This corresponds to how we, for example, search the World Wide Web using Altavista: the user states a number of words that describe the wanted document, and a list of documents ranked after presumed relevance is returned by the system.

Each group participating in the ad hoc task is given a large set of documents (approximately 2 GB) plus 50 topics. First, the participants have to produce a new query set from the topics given, thereafter these queries are run against the given collection. For each topic, a ranked list of 1 000 documents retrieved from the collection should be returned.

The document collection used for TREC consists mainly of articles from, for example: Wall Street Journal; Foreign Broadcast Information Service (FBIS); The Financial Times; Federal Register; and The LA Times. (For more on TREC, see Voorhees and Harman (1998b).)

2.2 The Relevance Judgement

In information retrieval, one way to measure how relevant a set of retrieved documents is, is by looking at *precision* and *recall*. By precision we mean the proportion of relevant documents in the retrieved set. Recall is the proportion of relevant documents in the whole collection which the system has retrieved. Both measures take a value between 1.0 and 0.0, where 1.0 corresponds to the best performance.

The measure used in the experiments described below, and one of the measures used in TREC is *mean average precision*. For a single query, the average precision is the mean of the precision obtained after each relevant document that is retrieved. By averaging this value for several queries, we obtain the mean average precision. This measure is sensible to the rank of the relevant documents: having the relevant documents at the top will result in a higher value, i.e., such documents will be given higher weights.

As the data collection is fairly large, as is the amount of runs submitted by the participating teams, it is not possible to calculate the exact precision or recall. Instead, a number of human judges take the top 100 for each query and mark the documents for

relevance. These judgements are used as a relevance pool for the rest of the retrieved documents. Hence, a rather large amount of documents will not be judged at all (the documents that are ranked as 101–1 000 and that have not been retrieved by anybody else on the first 100 places). However, if none of the participating systems have retrieved a document in the top 100, sampling tests seem to indicate that it is not very likely to be relevant.

2.3 Natural Language Information Retrieval

The GE/Rutgers/SICS/Helsinki team (further described in Strzalkowski *et al.* (1998)) has for several years participated in TREC with the aim to show that linguistic features can be useful for classifying documents as relevant or irrelevant to a query. To accomplish this, different indexing approaches, term extracting, and weighting strategies have been used to build a number of *streams*. Each stream represents an alternative text indexing method; some require complex linguistic processing, while others are based on simple quantitative techniques. The results obtained from the different streams are lists of documents ranked in order of relevance: the higher the rank of a retrieved document, the more relevant it is presumed to be. The ordering is based on the relevance score—a figure produced by the stream, reflecting the document's accuracy as judged by the system. The streams perform in parallel and the results from the different streams should be merged to produce one final result.

3 The Data Set

The experiments described in this article were performed with TREC-7 data, processed with methods developed for TREC-8. We merged four different streams; these will be described below. For more details on the indexing methods see Strzalkowski *et al.* (1999).

3.1 The Indexing Methods

As mentioned above, the different streams correspond to different indexing methods. In this section, we will describe these streams. For simplicity we will refer to the streams as *one*, *two*, *three* and *four* respectively.

Stream one Automatic expansion, i.e., passages extracted from retrieved documents are added to the query. Uses proximity phrases—that is two (or more) terms must be adjacent (or within a distance) of each other. Runs on *stem* stream (stemmed words and stop words removed).

Stream two Before expansion the terms from stream *one* were added to the topics using a feature in InQuery called #phrase operator (giving higher weight to co-occurring phrases) in case it is a phrase. Automatic expansion. Runs on *stem* stream.

Stream three Automatic expansion. Runs on *stem* stream.

Stream four Single words and head-modifier word pairs. The word pairs contain noun phrases but also other syntactic constructions which have similar meaning.

The words from the title of the topic description are repeated three times before the query is processed for all four streams, thus giving different weights to the different fields in the topic.

3.2 The Data

As stated in the previous section, we performed the experiments using four different streams. Each stream produced a list of 1 000 documents for each query, and as we had a total of 50 queries, the whole set added up to 200 000 documents. The relevance pool (i.e., all documents that were reviewed by the human judges) for TREC-7 consisted of 4 674 relevant and 75 671 non-relevant documents (in total 80 345).

In table 1 below, we will give some statistics about the streams: the number of relevant documents retrieved for each of the stream (rel.); average precision (av. pr.); the number of queries for which the stream had the highest number of relevant documents retrieved as compared to the other streams (best); and the highest and the lowest relevance score for each stream over the 50 queries (max. and min.).

Table 1: Statistics about the data. (For explanations of the abbreviations, see above.)

	one	*two*	*three*	*four*
rel.	2 779	2 604	2 918	1 435
av. pr.	0.2442	0.2266	0.2442	0.0918
best	15	11	9	3
max.	0.4729	0.4721	0.4765	0.4593
min.	0.4062	0.4052	0.4062	0.4036

It is interesting to note that although stream *three* retrieved about 5% more relevant documents than stream *one*, the average precision is the same for the two streams. One should also note that although stream *one* was the best stream on many more occasions than stream *three*, they had the same average precision, thus showing that the ordering of relevant versus non-relevant documents is important.

4 Experiments

In this section, we will first describe the merging algorithm used, and then look at the results we got when applying this algorithm to the task at hand. As mentioned previously, the data used was the TREC-7 data, and we also used the relevance judgements from TREC-7 to measure the average precision.

4.1 SEQUEL

SEQUEL is a merging algorithm developed by Asker and Maclin (1997) and has been used successfully for image classification tasks. The rationale behind SEQUEL is to find the most confident classifier down to a certain threshold. It requires that the list be sorted by—in

this case—relevance score. The confidence is calculated by finding the classifier with the highest proportion of correct classifications (i.e., relevant documents) at the top, down to the first non-relevant one. The threshold will be the lowest relevance score within this interval. The items covered by the span are removed from all classifiers.

The training was performed on 40 out of 50 queries—setting aside 10 for testing. Two different implementations of the algorithm were made: one where all queries were sorted by the judgement of the system; and one where the program examined the confidence for each query at a time, taking the average as the result. All non-judged documents were removed, leaving only the judged documents for consideration. In addition, one small set of experiments was performed where the ranking scores were normalised to fall between 1 and 0 (only for the first implementation).

4.2 Results and Discussion

Although the algorithm performed well, no merged list had higher average precision than the best individual stream (which is our criteria for a successful merging). For this reason, no diagrams or curves of these runs will be presented here, but we will conclude with some reflections of possible reasons why the algorithm did not work for this type of data. We will also suggest an alternative method, which still has to be realised before we can draw any conclusions about its ability.

The algorithm tended to favour the best performing classifier (it being the most confident stream) and discarded the additional information that the weaker streams may have contributed with. This could possibly be because the algorithm was not very forgiving: immediately upon finding an irrelevant document the stream was discarded. SEQUEL also tended to work with chunks of documents, covering too many at the time. This could be due to the fact that the relevance scores given by the retrieval systems range over a quite limited span. (The above mentioned normalisation of the scores did not, however, yield a better result.) A shortcoming of the algorithm is that it does not take a possible overlap of the retrieved documents in the different streams into account.

At a certain value the best performing classifier may be considered the default classifier, i.e., it can be used as the single classifier. While experimenting with different threshold values it turned out that the more we let the best performing stream influence the result, the better it got. That was simply because we had fewer of the less well-performing streams lowering the result.

This automatic classification scheme is seemingly tuned for a different type of task. Hence, we need to find more forgiving methods, which do not discard a stream immediately upon finding an irrelevant document: document relevance is a debatable issue in itself, and cannot easily be compared to other classification tasks where the errors are of a more clear-cut nature.

5 Future Work

SEQUEL was implemented to consider the top 1 000 for each query and classifier. This means that for the majority of the documents, we will only know the relevance scores

from one or two streams. It would be more appealing to apply a method where every document to a larger extent could benefit from the fact that we use several retrieval methods. Something that points in the direction that this could be a feasible approach is the fact that the ranking scores continue to be more or less at the same level even at end of the list of retrieved documents.

We will conduct a set of new experiments that will take into account the judgement of every stream: all documents that have been ranked among the first 1 000 documents by at least one stream out of N streams will constitute a "document pool" of at least 1 000 documents and not more than N*1 000 documents. We will let every stream score all documents in this pool. As a first experiment, we will implement a simple linear combination of the judgements from the four streams. By a weighted sum of all the scores from each of the streams we will get a total score that includes the knowledge of all streams in question. As mentioned before, the span for the ranking scores is not that large, and therefore even a very small score can alter the ordering of the documents. For these tests we will retain the non-judged documents, and we will experiment with both non-normalised and normalised scores.

A document pool for the four streams would constitute of the 113 135 unique documents (out of 200 000 in the retrieved set, meaning that the overlap is about 43%). Of these 3 408 are relevant. If comparing this figure to the 2 918 documents that stream *three* retrieved the other three streams could—at least in theory—contribute with 490 relevant documents. Below, in table 2, we will give an example from one query. Here we can see that there is a difference of 55 documents between the best performing stream (stream *one*) and the document pool. If including these additional documents in the best retrieved set, the number of relevant documents would increase with 32%.

Table 2: Number of relevant documents found to query no. 354.

one	*two*	*three*	*four*	*doc. pool*
172	142	170	115	227

There is a possibility that some documents could get a better total score by having been given four low scores (too low to be among the best 1 000 documents for any stream) than one with a high score on one and no rating on the other streams. However, if none of our streams ranks a document among the top 1 000 we will discard it. If the experiments with simple linear combinations turn out satisfactory—i.e., better than the non-adapted learning algorithms—we will continue with more sophisticated methods, for example by weighting the different streams. As a baseline, we will use the average precision we get when combining the scores, not looking beyond the top 1 000.

6 Related Work

An approach similar to the one described in the previous section was used by Mayfield *et al.* (1999) at TREC-8. The team used a linear combination with manually set weights for merging three different runs. The ranking scores from the runs were normalised to fall between 0 and 1. An improvement was obtained with the merging

compared to the single runs. However, only the top 1 000 was considered for each run, and when having conducted the above described experiments, we will be able to conclude whether our approach is more favourable.

References

Asker, L. and Maclin, R. 1997. Ensembles as a Sequence of Classifiers. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97)*, Nagoya, Japan.

Bartell, B. T., Cottrell, G. W., and Belew, R. K. 1994. Automatic Combination of Multiple Ranked Retrieval Systems. In *Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval (SIGIR'94)*, Dublin, Ireland.

Dietterich, T. G. 1997. Machine Learning Research: Four Current Directions. *AI Magazine*, **18(4)**:97–136.

Mayfield, J., McNamee, P., and Piatko, C. 1999. The JHU/APL HAIRCUT System at TREC-8. Preliminary TREC-8 Report. Gaithersburg: NIST.

Merz, C. J. 1999. Using Correspondence Analysis to Combine Classifiers. *Machine Learning*, **36(1/2)**:33–58.

Strzalkowski, T., Stein, G., Wise, G. B., Perez-Carballo, J., Tapanainen, P., Jarvinen, T., Voutilainen, A., and Karlgren, J. 1998. Natural Language Information Retrieval: TREC-7 Report. In Voorhees and Harman (eds.) (Voorhees and Harman, 1998a).

Strzalkowski, T., Perez-Carballo, J., Hulth, A., Karlgren, J., and Tapanainen, P. 1999. Adhoc experiments performed by the GE/Rutgers/SICS/Helsinki team in the context of TREC-8. Preliminary TREC-8 Report. Gaithersburg: NIST.

Voorhees, E. M. and Harman, D. K., editors. 1998a. *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242.

Voorhees, E. M. and Harman, D. 1998b. Overview of the Seventh Text REtrieval Conference (TREC-7). In Voorhees and Harman (eds.) (Voorhees and Harman, 1998a).

Extracting Keywords from Digital Document Collections

Anna Jonsson
Swedish Institute of Computer Science
Human Computer Interaction and
Language Engineering Laboratory
Kista, Sweden
E-mail: anjo@sics.se

Abstract

An indexing tool was built to provide for one of several information seeking tasks. In accordance with the basic principles of work held by the HUMLE laboratory at SICS, a solution regarding indexing would be a semi-automatic tool. This approach is also relevant as the continuation of the indexing project is conducted in co-operation with the Swedish Parliament, where a staff of professional indexers currently is investigating the utility of automatic and semi-automatic indexing tools to raise productivity.

1 Introduction

Digital libraries are complex information systems, which augment and extend traditional libraries by affording users better support for human problem solving and problem formulation. Digital libraries should be understood to be more than a haphazard collection of electronic resources and associated technical widgets for creating, searching, and using information in various media and over networks. They are, or should be, tailored to the needs and tasks of a group or several groups of users, and their functional capabilities should support the information needs and uses of those individuals and groups.

Digital libraries are both an extension and integration of existing information sources, and through the advent of new technology and adjustment of tried and familiar technologies, a completely new concept. While digital libraries typically improve certain aspects of traditional libraries, most often today they leave other aspects unaddressed, which will decrease their usefulness. Traditional information institutions not only make information resources available to the public, but actively select, collect, organise, and preserve them, engaging in numerous behind-the-scenes tasks seldom addressed, or taken for granted in their digital counterparts.

Despite recent advances in both computer technology and computational linguistics, retrieving and extracting useful information in large document collections is still very troublesome. Freetext search is certainly useful and fast, and generates a generous amount of results, but distinguishing the relevant documents from the non-relevant in the abundance of returned documents is a problem. Other systems for structuring information to enhance availability has traditionally been by storing information about

documents, books, and texts in bibliographic cards; and by indexing the documents by lists of keywords or keyphrases.

2 SICS Digital Library

To understand the context to our current indexing work, this chapter gives an overview of the theoretical background important in the Digital Library Project at SICS (DigLib), as well as a short description of the project itself.

2.1 Theoretical Background

The belief that one kind of information retrieval system, i.e. freetext search systems, can suffice for, and even replace other systems, and thus provide for all information seeking needs users have, is widely common today. This maybe due to the recent technological advances that have solved a considerable amount of the problems this area suffered initially, mainly regarding speed and storing possibilities. However, Belkin and others have analysed user strategies for information seeking (Belkin & Cool, 1993; Belkin, Cool, Stein & Thiel, 1995), and recently Belkin and Carballo (1998), based on extensive user observations, found that humans utilise a multitude of strategies in the task of information seeking. The four strategies people spend most time on are: *finding a known information object*; *recognising useful information objects by scanning through an information resource*; *evaluating the usefulness of information objects*; and *determining the content or structure of a collection of information objects*. The first of these strategies, "finding a known information object" can be seen as corresponding to bibliographic cards, containing information about author, title, publishing data, and possibly an abstract, used for example in traditional libraries. An example of what "recognising useful information objects by scanning through an information resource" means, can be the behaviour we adopt when browsing through web pages. The third strategy we spend time on, "evaluating the usefulness of information objects", is e.g. when we try to distinguish what documents are relevant to our query after a search, for example a search on Altavista. And finally, "determining the content or structure of a collection of information objects" is what we do when familiarising ourselves with a book by looking at the table of contents, or by looking up the keywords listed in the book's index.

So, the conclusion is that multiple information seeking strategies need to be met by several information seeking tools, which clearly indicates that a single information access tool would not be sufficient.

2.2 DigLib

SICS runs DigLib, a project for the study and application of digital libraries. The central issue for DigLib is focusing on usage: studying how and why users interact with document collections, and trying to build tools incorporating new technology to aid users in the tasks we find they try to solve. The leading principle is that general solutions to information access problems tend to be unsatisfactory, and that tailoring technology to

specific requirements of professional users is more rewarding than trying to automate general tasks for all.

To provide for the above-mentioned information seeking strategies proposed by Belkin and Carballo, a platform including different tools was built within DigLib. As a tool for "finding a known information object", we included Dienst, a bibliographic look-up engine. Dienst performs a search in bibliographic records, where, among other things, title and author are specified. It provides rapid, consistent, and predictable results: it indexes documents by a small number of highly relevant fields without bothering with the full document texts (Davis & Lagoze, 1994). "Recognising useful information objects by scanning through an information resource" was made possible through a systematic structure of HTML links, organised hierarchically by information source. The third prototypical information seeking interaction "evaluating the usefulness of information objects" cannot be met by adding a single tool, as this concerns different aspects of individual documents or sets of documents, and requires specific techniques for information refinement. Evaluating a document in regard to a user's information need, can for example be accomplished by comparing the document to other documents in the collection or by custom-made summarisations of retrieved documents.

There are several ways to support the fourth information seeking interaction style, "determine the content of a collection of information objects". Although graphical visualisation techniques using various metaphors show promise of usefulness, other solutions must be offered, since up to 64 percent of the population have difficulties using maps for orientation even in straightforward navigation tasks (Streeter & Vitello, 1986). We implemented the Keyword Extraction Function (KEF, see further in section 3.1 below), that extracts and presents keywords from documents in a way that resembles an index in a book - something most of us are familiar with, and many of us use for precisely the purpose of familiarising ourselves with the contents of a book.

3 Previous Work

Within the DigLib project we have, as outlined above worked on integrating several tools that meet different information seeking strategies. The tool we developed in-house corresponds to one of these strategies, and is described below.

3.1 The Keyword Extraction Function

The tool called the *Keyword Extraction Function* (KEF) was the first prototype to an indexing tool. The function takes all words from the text-files and applies a lexical filter, which selects all nouns from these texts. For this it is dependent on a part-of-speech tagger. We investigated the possibility of using other criteria for keyword spotting, such as word length; long words tend to be topic-specific, especially in a compounding language such as Swedish. The results were not completely discouraging, but we found that the benefits a tagger gave were not limited to term spotting, but included conflation of morphological variants - which in general is desirable. We concluded that tagging is necessary for term extraction.

Nouns were chosen as index words, as they seem more information dense than other word-classes, and are most often used in retrieval queries (Källgren, 1984; Källgren, 1992). Complex noun phrases were disregarded, after having analysed a sample set of documents. In addition Ingwersen (1992) states that automatic indexing techniques based on single words are quite effective, and multi-stream information retrieval experiments shows that single term retrieval in general is the single most effective knowledge source for information retrieval (Strzalkowski et al., 1997), compared to, among others, multi-word terms.

By selecting all nouns occurring in the texts for further inspection, KEF overgenerates terms. The assumption was that overgenerating terms and excluding non-relevant ones is a safer bet than attempting to pick only the most relevant ones. A number of researchers have in various ways shown that frequency or repetition in addition to lexical category is important for modelling term relevance (Luhn, 1959; Salton, 1989; Justeson & Katz, 1995).

Our conclusions from this work were that morphological and lexical tagging is necessary for term extraction and that a keyword index needs further techniques for refining the choice of included terms. Current experiments include statistically based term selection metrics, and the possibility of generating different index term lists for different purposes: a shorter list could conceivably be used for overview; a more exhaustive list for finding precisely which items to peruse further.

4 Current Work

There are today many organisations that daily deal with large amounts of documents, index them manually, i.e. create lists of keywords describing the document, in order to enhance the documents availability for information finders within the organisation as well as, in some cases, for the general public. Indexers sometimes have access to knowledge databases of some sort, often a structured ontology or knowledge model to aid them in their work. The quality of these manually produced indexes is high, however, although when having access to ontologies, humans do not seem to index in a consistent way (Earl, 1970, Kowalski, 1997), and the work is both time-consuming and expensive. As comparison, automatic indexing, which has undergone a dramatic change to the better since the beginning in the 50s and 60s, gives quick results that can be said to be consistently done, and it is far less expensive than manual indexing, but the quality is still questionable (Salton, 1989, Kowalski, 1997).

So, bearing in mind that manual indexing is of high quality but time consuming and inconsistent, and automatic indexing is fast, consistent but of lower quality - how can we improve the quality of indexing, and relieve pressure on those who are professional indexers, and increase productivity?

4.1 Principles of Work

As mentioned above, the leading principle when working with DigLib, was to avoid general solutions. In our case, and as an answer to the question ending the previous section, the solution to us is semi-automatic tools. To understand why, a description of some of the basic principles important for the work done at the HUMLE laboratory might be necessary. These principles concern: system context; methods for design; and the development of tools for professionals.

Firstly, it is important to adapt a system to an existing situation, understanding the language used in some specific context, and to include modality-specific information in the system specification. There is often no best general solution available, but a system needs to be customised to fit in the workflow of the organisation. Secondly, it is important to bear in mind the difficulty involved when trying to collect and collate the right sort of information; information that has bearing on the design process. That is, finding a method of relating information about a workplace and the individuals therein to the design process is essential. And thirdly, the professional user shall be aided in the task at hand, and it should be an intelligent aid system. Complete automation should be regarded with some degree of scepticism. The goals are to raise productivity, efficiency, and quality of information work.

4.2 The Swedish Parliament

One of the organisations we co-operate with regarding indexing is the Swedish Parliament. They index large numbers of documents yearly, in order to make them accessible both to information specialists and to the general public, and their work on this plays an important societal role. This work has progressed for a long period of time, during which they have developed an extensive hierarchically organised domain specific thesaurus (or knowledge base). Figure 1, below, shows an excerpt of the thesaurus: the word *arbetshandikapp* (work disablement) and it's broader term (BT), narrower terms NT, related terms (RT) and a description of what the word means (SN).

```
Arbetshandikapp
    BT Arbetsliv
    NT Arbetsbiträde
    NT Näringshjälp
    NT Skyddat arbete
    RT Anställningsfrämjande åtgärder
    RT Handikapp
    RT Lönebidrag
    SN Nedsatt arbetsförmåga pga fysiska, psykiska, förstånds-
       mässiga eller socialmedicinska handikapp - däri inbegripet
       missbruk av alkohol eller annat berusningsmedel.
```

Figure 1. Excerpt from the thesaurus developed at the Swedish Parliament.

As a consequence of the parliament's long history of manual indexing, radical departures from tradition is very much undesirable. Meaning that an aiding indexing tool in essence must function as a re-implementation of the tasks currently performed.

4.3 Towards a Domain Specific Indexing Tool

The development of the new indexing tool takes a slightly different starting point than KEF. The input comes in the form of a list of keywords generated by means of standard tf*idf calculations. As before, we have concentrated on nouns as keywords, at least as an outset. The new approach, from our point of view, is making use of the thesaurus, (mentioned above), for shortening the list of keywords. By aggregating several occurring narrower terms, the tool can then suggest to the human indexer, with some measure of confidence, a broader term that describes the document on a higher level of abstraction. For example, if the list of keywords shows that the document contains several occurrences of specific types of banks (e.g. Affärsbanker, Föreningsbanker, Investeringsbanker, Sparbanker) the indexing tool will suggest their broader term *Finansinstitut* (Finance Houses) to describe the document. The confidence measure will reflect the coverage of the terms occurring in the document that are corresponding to the terms listed in the thesaurus.

In order to improve the performance of the indexing tool, automatic evaluation based on previous manual indexing will be implemented. The manual indexing done at the Swedish Parliament over many years has been stored and used for an enhancement of the indexing accuracy. This will be taken into account also for our indexing tool.

Another facet that will be taken under consideration in the further development of the tool is the temporal aspect. Term meanings change over time, and index terms shift over time. This should be made visible to both the indexer and the viewer/user of the finished index. When e.g. looking up a new term, it may be relevant to also retrieve documents covering the same concept, although it might be described using slightly different terms.

In the future we want to enable the indexing tool to recognise when new terms should be added to the thesaurus. If there seems to be a new term frequently occurring in a number of documents indexed with the aid of the tool, it should make the manual indexer aware of this, by recommending that the term be added to the thesaurus.

5 Discussion

This article has given the settings to the current work performed within the DigLib project at SICS. The theoretical background is that people use various strategies when seeking information (Belkin & Carballo, 1998), and the strategy we have focused on is one way of allowing users to familiarise themselves with an information resource. This is accomplished by presenting an index of the document. The work started with the Keyword Extraction Function, implemented in SICS' Digital Library platform (Hulth & Jonsson, 1999), and has later on developed into the semi-automatic indexing tool currently under development.

Semi-automatic indexing may not be too hard to accomplish per se, as it is a rather straightforward task. Using the semi-automatic indexing in combination with ontologies may, however, be more of a challenge, especially the question of how to combine two or more differently structured knowledge models, and allow them to communicate and

co-operate with each other. This will be one focus of research that we will be experimenting with in the near future.

Another focus is the question of what makes a good index. There is more to indexing than merely making lists of words, whether they have frequency measures and weights of various kinds attached to them or not. What criteria do humans use when indexing a document for example?

When we have realised the implementation of the tool, and evaluated its performance in the authentic environment, adapted to the requirements of the workplace, we will know to what extent our presumptions regarding semi-automatic indexing were correct.

Acknowledgements

A lot of the previous work was performed in co-operation with Anette Hulth.

References

Belkin, N.J. & Carballo, J.P. 1998. Understanding and Supporting Multiple Information Seeking Strategies, a TIPSTER Phase III Research Project. URL: http://www.scils.rutgers.edu/tipster3/18month/index.htm

Belkin, N.J., Cool, C., Stein, A. & Thiel, U. 1995. Cases, Scripts and Information Seeking Strategies: On the Design of Interactive Information Retrieval Systems. Expert Systems with Applications.

Belkin, N.J., Marchetti, P.G. & Cool, C. 1993. BRAQUE: Design of an interface to support user interaction in Information Retrieval. *Information Processing & Management Vol 29*, 3, 325-244.

Davis, J.R. & Lagoze, C. 1994. A Protocol and Server for a Distributed Digital Technical Report Library. URL: http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR94-1418?a

Earl, L.L. 1970. Experiments in Automatic Extracting and Indexing. *Information Storage & Retrieval. Vol. 6, pp. 313-334.* Pergamon Press.

Hulth, A. & Jonsson, A. 1999. *An Experimental Digital Library Platform – A Demonstrator Prototype for the DigLib Project at SICS.* Stockholm. SICS Technical Report T99:02.

Ingwersen, P. 1992. *Information Retrieval Interaction.* Taylor Graham Publishing.

Justeson, J. & Katz, S. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering 1 (1) 9-27.* Cambridge University Press.

Kowalski, G. 1997. *Information Retreival Systems: Theory and Implementation.* Kluwer Academic Publishers.

Källgren, G. 1984. *Automatisk exerpering av substantiv ur löpande text. Ett möjligt hjälpmedel vid datoriserad indexering?* IRI-rapport 1984:1. Institutet för Rättsinformatik. Stockholm University. (In Swedish.)

Källgren, G. 1992. *Making maximal use of surface criteria in large-scale parsing: the MorP parser*. Papers from the Institute of Linguistics. (Publication 60). University of Stockholm.

Luhn, H.P. 1959. Auto-Encoding of Documents for Information Retrieval Systems. In: Boaz, M. *Modern Trends in Documentation*. (Ed.). London: Pergamon Press. (Reprinted in: Schulz, C.K. (1968). *H.P. Luhn: Pioneer of Information Science, selected works*. (Ed.). New York: Sparta.).

Salton, G. 1989. *Automatic Text Processing*. Addison-Wesley Publishing Company.

Streeter, L. & Vitello, D. (1986). A Profile of Drivers' Map-Reading Abilities. *Human Factors, 1986, 28(2), 223-239.*

Strzalkowski, T., Guthrie, L., Karlgren, J., Leistensnider, J., Lin, F., Perez-Carballo, J., Straszheim, T., Wang, J. & Wilding, J. 1997. Natural Language Information Retrieval: TREC-5 Report. *Proceedings of the fifth Text Retrieval Conference*, Donna Harman (ed.), NIST Special Publication, Gaithersburg: NIST.

Ontologically Supported Semantic Matching

Atanas K. Kiryakov, Kiril Iv. Simov
Linguistic Modelling Laboratory
Bulgarian Academy of Sciences
Acad. G. Bontchev Str. 25A, 1113 Sofia, Bulgaria
diogen@diogenes.bg, kivs@bgcict.acad.bg

Abstract

Evaluation of the closeness of two texts is a subtask for *FTR* and *IR* systems. The basic means used to accomplish it is the matching of *atomic text entities* (ATEs) such as words, stems, simple phrases and/or concepts. We address the question how concepts can be used as ATEs more efficiently in order to match *"small duck"* with *"small bird"*. The *onto-matching* technique introduced in the paper makes extensive use of lexical ontologies similar to WordNet.

We work with two tasks in mind: query expansion and text concept indexing. We outline some arguments showing why onto-matching is useful and how it can be implemented. Also, we conducted some experiments with query expansion for AltaVista.

1 Introduction

"A typical information retrieval task is to select documents from a database in response to a user's query, and rank these documents according to relevance." Strzalkowski et al (1998). The relevance must be defined on the basis of the concepts represented in the text and in the query. Usually information retrieval (IR) systems calculate the relevance of a text with respect to some query according to the number and the profile of the occurrences in the text of some elements from the query. The main stream of research in IR is towards the development of methods for the recognition of more meaning bearing elements of texts which can then be used to evaluate the closeness of the two texts (queries are also texts).

Most often a document is converted into a bag of words, stems or other textual elements which we call *atomic text entities* (ATEs) (sometimes information associated with them is also used). The hope is that these elements explicate the concepts represented by chunks of text and so define the topics of the document. Similarly, the query is considered to be itself a bag of words, stems, etc. and again the hope is that they explicate the concepts of the query.

Although words **denote** concepts, often they are not sufficient in themselves to **explicate** these concepts. They can be thought of as names for the concepts in the world. Usually the definition of a concept spells out what are the constraints on its possible representatives or instantiations, it could also give some prototype

information and information about this concept's relationship to other concepts. It is our opinion that users of information retrieval systems rarely search simply for words. Rather, they are interested in the concepts that words represent. Thus concepts (including at least some parts of their definitions and relations to other concepts) should be included amongst the atomic text entities. In this way we will capture the intuitive expectation that when one is searching for *bird* the occurrence of *duck* is also relevant. This is so because the word *duck* represents a subconcept (more specific concept) of the concept represented by *bird*.

The problem of the word-to-concept correspondence is well known and intensively studied in a number of areas like linguistics, psychology, artificial intelligence, etc. In order to demonstrate some of its aspects we give here a small example. Let us consider the following top-ontology of particulars (taken from Guarino (1998)):

```
Particular
    Location
        Space (a spatial region)
        Time (a temporal region)
    Object
        Concrete object
            Continuant (an apple)
            Occurrent (a fall of an apple)
        Abstract object (Pythagoras' theorem)
```

Here, objects are considered to be *concrete* because of their ability to have some location. *Continuants* are what is usually considered to be *objects*, while *occurrents* correspond to *events*. *Continuants* have a location in space. They have spatial parts, but they have neither a temporal location nor temporal parts. *Occurrents* are "generated" by continuants, according to the way they behave in time. Occurrents always have other occurrents as parts (continuants take occurrents as parts, but are not part of them). They have a unique temporal location, while their exact spatial location can not be defined in the general case. *Abstract objects* do not have a location at all. Most of the entities classified as abstract objects can also be thought of as *universals*.

Depending on the definition of a concept and therefore on the objects this concept denotes it can be classified under one or another branch of this ontology. Thus concepts *lexicalized* via words in a natural language (or *lexical concepts*) will belong to different branches of any ontology extending on the above minimal ontology. For example, the English word *book* denotes at least the following concepts: "information unit", "physical object" and "commodity" which belong to different branches. As a physical object *book* is a continuant and as an information unit it is an abstract object.

One another important point is that world knowledge, that is our repository of concepts and facts, is considerably more massive than is the set of lexicalized concepts. Therefore, if we use only words as concept denoting entities, we can hope to find only a fraction of the concepts that we have available to us.

In this paper we investigate the possibility to use WordNet (see Fellbaum (1998)) as a source for the explication of some concept relations in order to improve the matching of ATEs in documents and queries. More specifically, we exploit the

hypernym-hyponym relation. We call this augmented matching of concepts — *onto-matching.* This improvement can be used in the core of both *FTR* and *IR* systems, as well as in other places, like information filtering, dictionary look up, information extraction, etc.

The structure of the paper is as follows: the following section gives an overview of WordNet and some of the approaches to using WordNet to enhance the precision in IR systems; afterwards, we discuss different approaches to "concept" search in texts and we introduce the central notion of the paper — onto-matching; the next section is devoted to the application of WordNet and onto-matching to query expansion and document indexing; the last section concludes the paper and lists some problems and directions for future research.

2 Using WordNet to Enhance IR

2.1 WordNet — a lexical ontology

The following is a concise description of WordNet as given by its developers: "WordNet is an on-line lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets." see Miller (1995) and Fellbaum (1998). Some other basic relations are hyponymy, hypernymy and meronymy for nouns, entailment for verbs, antonymy for adjectives. The relations are divided in two levels: conceptual relations that connect synonym sets such as the hyponymy-hypernymy relation and lexical relations that connect particular words in synonym sets such as the antonymy relation.

The fundamental building block of WordNet — the *synonym set* or *synset* represents a lexical concept via the set of words (in some cases also phrases, idioms or collocations are used) that lexicalize this concept. Besides these "names" (and informal glosses) no other information is given about the concept, i.e. there are no formal definitions or prototype information. The main design principle of WordNet is to situate lexical concepts in semantic nets constructed with respect to a number of semantic relations between concepts (or words) that are sufficient to discriminate between them. This design principle implies the division of WordNet into four non-interacting semantic nets — one for each open word class. This separation is based on the fact that the appropriate semantic relations between the concepts represented by the synsets in the different parts of speech are incompatible. For example, there is no semantic relation that would appropriately connect a verb synset with a noun synset and that would make a reasonably detailed distinction between these synsets and other verb and/or noun synsets.

Additionally, the semantic nets in WordNet are divided in subnets by the so called unique beginners which determine hierarchies of mutually incomparable lexical concepts. These unique beginners play a role similar to that of the ontological classes given in the above top-ontology. The following synsets define some of the unique

beginners for nouns:

{act, activity}	{animal, fauna}	{artifact}
{cognition, knowledge}	{natural object}	{possession}
{process}	{quantity, amount}	{shape}

The structure and the content of WordNet determine the ways and the extent to which it can be used to explicate the concepts in a text. Most profitably, one can use WordNet to determine the lexical concepts designated by a word and their relations to other lexical concepts.

2.2 WordNet and IR projects

WordNet was used in several projects to enhance the precision of the search for relevant documents. These include (among others): Voorhees (1998), Guarino et al (1999) and Gonzalo et al (1998). Gonzalo et al (1998) uses WordNet to index texts in two ways: first, they attach to each word its sense, using an index of three numbers — one for its part of speech, one for the unique beginner within this part of speech and a third one pointing to the word-sense in this file; second, they attach to each word the right synsets (lexical concepts). Then they use the standard vector based matching of the query to the documents using the added information. The experimental work shows that the performance of document retrieval by summaries improved by 29%!

Voorhees (1998) reported on two different tasks: word-sense disambiguation as part of the problem of conceptual matching and semantic expansion of the query. The conceptual matching experiment failed, because of a wrong strategy for automatic disambiguation combined with an extremely error-sensitive relevance evaluation method — the extended vector space model. The idea, in itself, is much like the one employed in Gonzalo et al (1998), where they studied the sensitivity of concept indexing against disambiguation errors and reported good results despite the 30% of errors. The goal of the second experiment was query expansion on the basis of lexical relations encoded in WordNet. All kinds of relations were studied as possible directions for the expansion with limited or unlimited transitivity. The results reported, however, concern only the case where all kinds of relations were traced for just one step. The conclusion was that such an expansion will lead to some improvements in the case of relatively short queries.

The OntoSeek project (Guarino et al (1999)) performs knowledge extraction with the support of the Sensus ontology (Knight & Luk (1994)). As a lexical front-end it uses WordNet and then maps the synsets to the formal concepts in Sensus. The goal is to provide means for knowledge acquisition from a knowledge base of lexical conceptual graphs (LCG). The target domains are on-line product catalogues and yellow pages. The results are descriptions of products or companies that can be matched with queries while taking into account ontological dependencies. This approach, however, presupposes semi-automatic encoding of the descriptions into a special form.

2.3 Including hypernymy in the retrieval

In our work we investigate the use of a more complicated concept matching approach augmenting some aspects of the approaches mentioned above. In our view a concept in a text is defined not only on the basis of its synset (taken as index) but also on the basis of other semantic relations, especially hyponymy and hypernymy.

We envisage two tasks: *Query expansion.* We expand the query by adding the hyponyms of the words it contains. Such an expanded query is evaluated with respect to documents for which no concept indexing has been done (using AltaVista for instance); *Concept indexing.* The texts of the documents are extended by a bag of concepts mapped to their words. These concepts are determined on the basis of the hypernymy-hyponymy relation.

3 Concept search

Concept search is defined in terms of atomic text entities which are extracted from texts and which are used as units in the evaluation of their closeness. In the usual query-document scenario we talk about query reengineering or expansion, while processing of the documents can be thought of as some sort of indexing. After the two texts (a document and a query) have been appropriately processed the sets of detected ATEs are matched with one another. There are also "stop-ATEs" which are defined in such a way that from each set of detected ATEs some of the entities are deleted. Usually, the deleted entities are those that denote overly broad concepts that would be found in any text. In our work we parametrize the notion of "stop-ATEs" to depend on the context and the wish of the user.

Most of the approaches to information retrieval we are aware of use the standard vector-space model to match the ATEs in texts and evaluate the semantic distance. The following is an overview of some of these approaches and the ATEs they use:

- *Word-stem.* With the help of an inflectional or a derivational morphological analyzer each word in the text is converted to its stem. The set of stems is used as an index space over which the matching algorithm operates. For instance, all occurrences of "read", "reads", "readable", "reader", "reading" are mapped to the stem "read". The idea is that a family of morphologically related words represents a concept and each member of the family denotes just some of aspect of it.

- *Word-sense.* This approach presupposes the availability of a lexical database listing a number of senses for each word. Using a word-sense disambiguator the appropriate sense is attached to each word in the text. The set of word-senses is used as an index space in the same way as in the word-stem approach. This approach is reported in Gonzalo et al (1998) where an index pointing to the word-sense is attached manually to the words in the test set of texts. The following example is taken from Gonzalo et al (1998): the occurrences of "debate" are represented by "debate%1:10:01::" where the three figures index is pointing to the sense number in the corresponding file of WordNet.

- *Lexical concept.* This approach uses a lexical database which relates each word to its corresponding lexical concepts. Each word in the text is substituted by an appropriate lexical concept. The index space here is the set of lexical concepts. In this approach different words can share the same lexical concept. For instance, in Gonzalo et al (1998) lexical concepts are represented by the synsets' identifiers from WordNet. Thus "debate" is substituted by "n04616654".

- *Ontological chunks.* Here the lexical concepts attached to the words in the text are augmented by their super and subconcepts. Thus each word is substituted by a chunk of an ontology which determines its place in it. Some of the ontological chunks will share their top parts — some lexical concepts in the text will have the same superconcepts. The index space is more complicated because we have to account for the ontological relations in the chunks. It is this approach that we investigate in this paper.

In the first three cases we can claim that the index spaces consist of points (word-stems, word-senses, lexical concepts). The matching algorithm has to compare these points in order to evaluate the matching of two texts. See Fig. 1 for a picture of this kind of matching. When the index space consists of ontological chunks, the matching algorithm has to be modified in an appropriate way to reflect the super/subconcept relation and the fact that concepts, even though they are not equivalent, could be considered relevant in the context of a certain retrieval task. This modification of the matching algorithm we will henceforth call **onto-matching**. See Fig. 2 for a picture of onto-matching.

The onto-matching approach is one attempt to overcome a certain intuitive asymmetry in users' expectations. For instance, in the case of *IR*, using the query/document schema, if one puts a more general query then all the documents that are evaluated as similar or more specific with respect to the concepts in the query are considered to be relevant. More general documents will be classified as irrelevant. When evaluating the relevance disregarding the structure of the texts, the same direction of generalization is expected for the atomic text entities (*ATEs*). For example, in most cases, a document containing only *bird* will be irrelevant to a query asking for *duck*. But under certain relevance evaluation schemata, a matching against the natural flow of generality can also be used (when in the document the superconcept is used in phrases that additionally constrain this superconcept and make it more specific).

Onto-matching gives more flexibility in the formulation of the query. The user can be additionally consulted so as to determine more exactly the content of the ontological chunks that are attached to the query. Depending on the settings, normally, the query will be indexed either by lexical concepts only or by lexical concepts and their subconcepts. In addition, one can direct the search using chunks that include also some of the neighboring concepts. This can be done by going up a few steps in the hierarchy to concept C and taking all subconcepts of C on the level of the lexical concept that was found in the text. Or to go on with our example, if in the text the concept *duck* is recognised, we go one step up in the hierarchy and take the immediate subconcepts. Then we also search for *goose* (this is done on the basis of

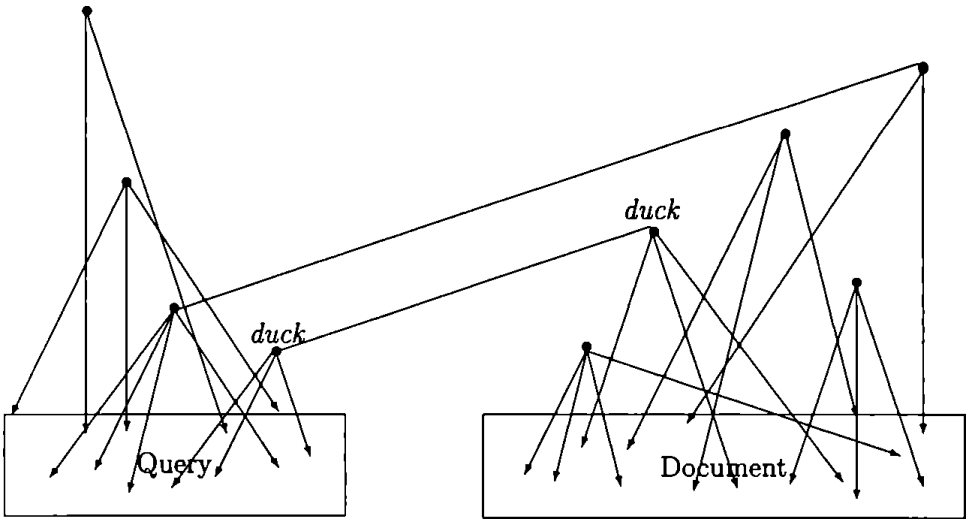


Fig. 1. Point to point matching. The dots represent the ATEs, the vectors represent the indexes to the occurrences of the ATEs in the text, the lines between the dots represent the matching between the ATEs in the query and in the document. For instance, the ATE for "duck" in the query matches the ATE for "duck" in the document.

the WordNet hierarchy). This can be done without the user's intervention if in the search engine an operator "SIMILAR" is defined that is doing this job automatically.

Query expansion with ontological chunks can be useful also when the document collection with respect to which it will be evaluated is not indexed even by lexical concepts. In this case, we have to attach to the lexical concepts in the query their subconcepts (or the words that represent them). This approach can lead to generation of hundreds of alternatives just for one of the words in the query. For example, trying to get the transitive hyponym expansion of the synset for *bird* (the first one listed in WordNet), we will end up with more than one thousand synsets representing more specific concepts.

One way to control onto-matching is to "refine" and strip the ontological chunks by removing some of the concepts which are not relevant. Such judgement can be made because of irrelevancy to the users' goals in particular run. Or, because of the nature of onto-matching we might want to exclude some "artificial" concepts or other "noisy" patterns that can be recognized in the ontology. For example, if we are searching for *bird* it can be the case that we want to exclude some of the branches of the hyponyms like *seabird*. These refinements in onto-matching can not be made before the actual search because they depend on the users' goals.

The mechanisms proposed so far serve as a reduction of the problem of matching between relevant but non-equivalent concepts. The goal is to make possible onto-matching with minimal complication of the currently used algorithms. The ontological chunks explicitly represent the necessary inferences and they are taken into account by the standard matching algorithm.

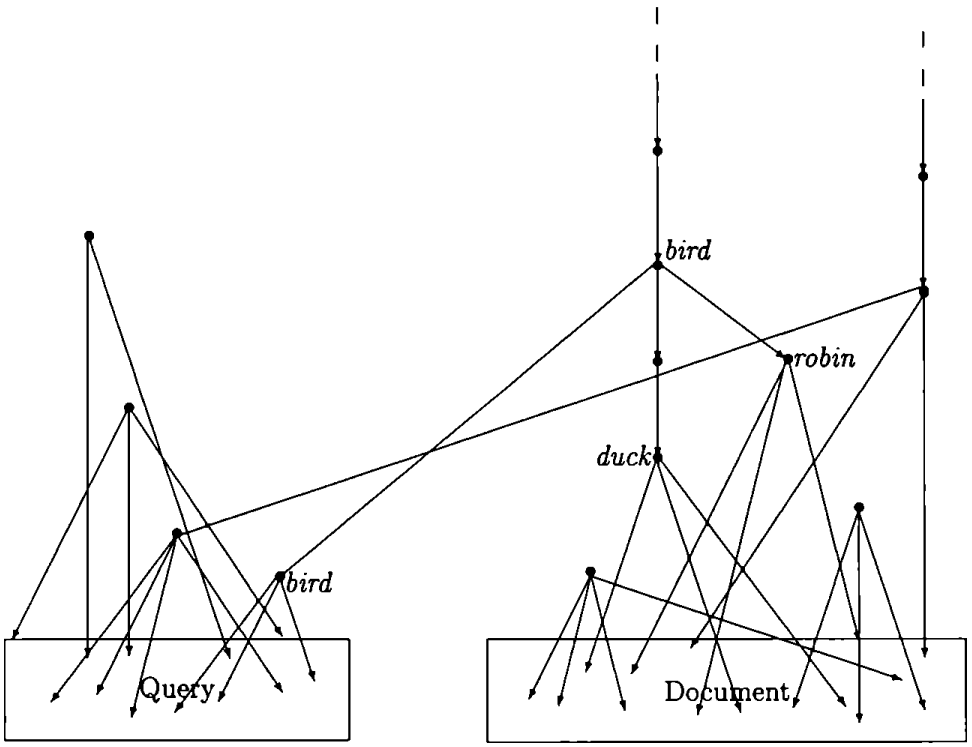


Fig. 2. Onto-matching. The dots represent the lexical concepts, the vectors represent the indexes to the occurrences of some lexical concepts in the text, the lines between the dots represent the matching between the lexical concepts in the query and in the document. Some lexical concepts are elements of the ontological chunks and the vectors connect them to their subconcepts instead of pointing directly to positions in the text. For instance, bird in the query matches duck and robin in the document via the ontological chunks above duck and robin.

4 WordNet for onto-matching

In this section we give some more concrete examples of the onto-matching approach using WordNet as the source for ontological chunks. We used the hypernymy-hyponymy relation between synsets for text indexing and query expansion. We conducted some experiments evaluating query expansion in searching the Internet. In both cases we presupposed that the texts were disambiguated and each word in them is connected to the right synset from WordNet. Some of the problems related to this assumption are commented in Section 5. In what follows we describe work with respect to the nouns in the text.

4.1 Text Indexing

Our goal is to index the text by the concepts corresponding to the words in it. Additionally, each lexical concept of a noun is indexed by the hypernym synsets.

All content words in the query text are indexed by their synsets only. Suppose a document contains an occurrence of the word *duck* to which the correct synset has already been assigned:

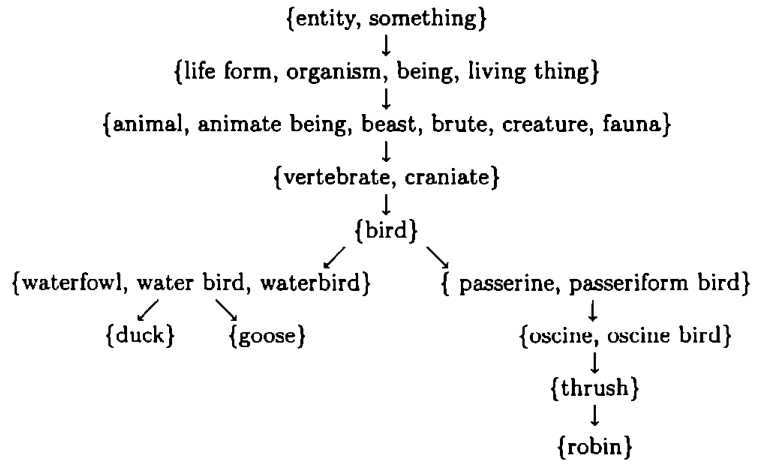
```
duck -- (small wild or domesticated web-footed broad-billed swimming bird ...)
```

The hypernyms for this synset are:

```
duck
   => anseriform bird
     => waterfowl, water bird, waterbird
       => aquatic bird
         => bird
           => vertebrate, craniate
             => chordate
               => animal, animate being, beast, brute, creature, fauna
                 => life form, organism, being, living thing
                   => entity, something
```

We index the occurrence of *duck* in the document with its synset and the synsets corresponding to its hypernyms. We call this **onto-indexing**. When a query contains a concept for *bird* it will be matched to the occurrence of *duck* (indexed with its hypernyms) without extending the query to the subconcept of *bird*.

One drawback of onto-indexing is that the overall size of the index will increase. We can partially solve this problem in two ways. First, we can reduce the number of the superconcepts deleting those that are not, strictly speaking, lexical concepts like "aquatic bird" in the hypernym chain above (see the Conclusion section). Also, the user can define concepts that should be excluded from the hypernym chain because they are specific to some domain of usage. In the example above one such lexical concept is "chordate" which is scientifically correct, but not much used in everyday life. Of course, if the search is for scientific documents then this concept should be retained and others excluded. Second, we can construct the ontology of the text so that hypernyms shared by some nouns in the text are represented only once. Suppose that in the text we have occurrences of *duck*, *goose* and *robin* and that *anseriform bird*, *aquatic bird* and *chordate* are excluded from the hypernym chains. In this case the index will look as follows:



Thus reducing the number of the added superconcepts we hope that the increase of the index will be logarithmic to the size of the lexical concepts found in the text.

4.2 Query Expansion

Here we assume that a query is matched against a collection of documents that are indexed only by words or stems. We then use WordNet to generate a list of their synonyms and hyponyms. This list is added in an appropriate way to the original query and then the actual matching is done. This is the approach employed in our testbed, but we should mention that ambiguous words (synonyms or hyponyms) can lead to a sharp decline of precision.

We carried out some experiments with a query expansion in order to estimate the applicability of onto-matching for the retrieval of documents from a heterogeneous set (web-pages from AltaVista) with a short query. The query in this case is not a normal text but resembles a formula constructed from words and operators like `AND`, `OR`, `NEAR` and others. The words in the query were mapped manually to the correct synsets in WordNet. Then the full set of synonyms and hyponyms for each noun was constructed. This set was added to the query, with the exception of the multi-word phrases.

For instance, for the query "`+hotel NEAR +cheap NEAR +London`" we expanded the noun "hotel". The corresponding synset in WordNet is:

```
hotel -- (a building where travelers can pay for lodging
          and meals and other services)
```

This synset has the following hyponyms:

```
hotel
       => hostel, hostelry, inn, lodge
           => caravansary, caravanserai, khan, caravan inn
           => imaret
           => roadhouse
       => motel, motor hotel, motor inn, motor lodge, tourist court, court
       => resort hotel, spa
```

We added the hyponyms to the query connecting them to the expanded word with `OR`. In the expansion process we exclude the phrasal synonyms like "motor inn". After the expansion the query became:

```
   (hotel OR
            hostel OR hostelry OR inn OR lodge OR
                        caravansary OR caravanserai OR khan OR
                        imaret OR
                        roadhouse OR
            motel OR court OR
            spa)
    NEAR +cheap NEAR +London
```

AltaVista returned 104 documents for the original query, against 138 for the expanded one. Experiments concluded with different queries confirmed the expectation

that the precision after the query expansion without onto-indexing is quite sensitive to ambiguous synonyms and hypernyms like "court" in the example. We concentrated on queries that do not contain highly ambiguous words in the expansion in order to get some approximation for the case of onto-matching.

We checked the precision for queries that return relatively small amount of documents and the general observation is that the expansion of the query did not depress it. The average increment of the recall is 30%.

In these experiments we were limited by practical considerations: we don't have at our disposal a collection of disambiguated documents indexed by lexical concepts; also, the majority of documents available for searches are not indexed by lexical concepts. Despite these practical constraints and the simplicity of the experiments, we can conclude that onto-matching is a promising approach for improving the precision in IR.

5 Conclusion

We found the results of the experiments encouraging, but have to point out a number of problems and directions for further research. The main difficulty is word-sense disambiguation. Throughout the paper we assume that each word in the text is correctly connected with the respective lexical concept. One could envisage a solution of this problem based on the use of semantic concordances in combination with statistical techniques similar to those used for POS-tagging.

Another problem is the recognition of multi-word concepts. For example, `water bird` is itself a concept and it will be strange to expand it in a query `bird` to something like:

```
water (bird OR cock OR hen OR eagle OR ...)
```

The right analyses of such terms will improve onto-indexing by attachment of the correct concept to multi-word terms in documents. This topic is described in details in Strzalkowski et al (1998).

If we have a more complicated concept representation where not only lexical concepts of nouns are used but also some additional constraints from the context are inferred (e.g. some attributes and their values) then a more sophisticated indexing mechanism will be needed. In this respect one can use the idea mentioned in Miller (1998) and compile for each noun a set of more appropriate attributes and their values (see pp. 40–41 there). These sets can be used for recognition of multi-word terms and for word-sense disambiguation.

In a more practical vein, we envisage to undertake experiments in onto-indexing over a collection of documents following the methodology of Gonzalo et al (1998) by manually attaching the appropriate chains of hypernyms to the words in the collection. These will give more reliable evidence for the usefulness of the ideas presented in this paper.

6 Acknowledgments

The research reported here was carried out within the Tübingen-Sofia International Graduate Programme in Computational Linguistics and Represented Knowledge (CLARK) funded by the Volkswagen-Stiftung. We would like to thank the Seminar für Sprachwissenschaft, Tübingen, for hosting the first phase of the writing of the paper. Also, we would like to thank Gergana Popova for her invaluable help. The first author is grateful to his colleagues from Sirma AI Ltd and Instill Corp. for their patience and understanding.

7 References

Gonzalo, J., Verdejo F., Chugur I. & Cigarran J. 1998. Indexing With WordNet Synsets Can Improve Text Retrieval. *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems.* Montreal.

Guarino, N. 1998. Some Ontological Principles for Designing Upper Level Lexical Resources. *Proceedings of First International Conference on Language Resources and Evaluation.* Granada, Spain.

Guarino, N., C. Masolo and Vetere G. 1999. OntoSeek: Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs. *IEEE Intelligent Systems.*

Fellbaum, C. 1998. *WordNet: an electronic lexical database.* (editor) MIT Press.

Knight, K. & Luk S. 1994. Building a Large Knowledge Base for Machine Translation. *Proceeding American Association of Artificial Intelligence Conference (AAAI-94)*, AAAI Press, Menlo Park, California. 773-778.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of ACM*, 11, 39-41.

Miller, G. A. 1998. Nouns in WordNet. *WordNet: an electronic lexical database.* (editor) MIT Press.

Strzalkowski, T., Guthrie L., Karlgren J., Leistensnider J., Lin F., Perez-Carballo J., Straszheim T., Wang J. & Wilding J. 1998. *Natural Language Information Retrieval: TREC-5 Report.*

Voorhees, E. 1998. Using WordNet for Text Retrieval. *WordNet: an electronic lexical database.* (editor) MIT Press.

Automatic Detection of Lexicalised Phrases in Swedish

Janne Lindberg
Dept of Linguistics
Stockholm University
beb@ling.su.se

I will present a system under development, called LP-DETECT. The system detects and analyses Swedish lexicalised phrases (LPs) in order to enhance subsequent parsing. LPs are one of a number of stumbling blocks related to word sequences that must be dealt with when parsing unrestricted text. LPs include semantic idioms, syntactic idioms and morphological idioms and so called valency breaking LPs. The system reported on consists of an LP lexicon of some 8000 LPs with analyses, a detection program written in perl and rules for disambiguating between and discarding LP analyses. A small evaluation of the system is also presented.

1. Lexicalised phrases

LPs are expressions that belong to the lexicon and consist of more than one word. Included are **semantic idioms** the meaning of which is not built up compositionally from the individual meanings of the words in the idiom. (An English example is *kick the bucket* meaning 'die', a Swedish equivalent *ta ner skylten* (lit. take the sign down) also meaning 'die'.) They should be lexically listed because of semantic reasons. Another group consists of **syntactic idioms** containing "ungrammatical" or non-standard combinations of words, in syntactic terms *(inte så värst* (lit. not so worst) meaning 'not particularly'; English example: *by and large).* The syntactic idioms do not make up regular grammatical structures and must therefore be listed as wholes in a parsing system. Of course, since syntactic idioms are syntactically irregular, no general function of semantic interpretation can apply over them and therefore a syntactic idiom is also a semantic one.

A third group consists of phrases containing words that are unique to the phrases where they occur, often lexical relics, for instance the Swedish LP *med nöd och näppe* meaning 'with difficulty'. The word *näppe* is not used outside this phrase and should therefore not be individually listed in a word lexicon. Related to the latter group are also phrases containing foreign words *(anno domini)* or nonce forms *(hux flux).* These could be called **morphological idioms** and they of course qualify as semantic idioms too.

In addition there are a large number of LPs, partly overlapping with the LP types described above, which I call **valency breaking LPs**. They simplify the phrase structure of sentences when detected as LPs. They tend to end in function words. An example of these is multi-word prepositions, i.e. *in spite of* (Swedish example: *på grund av*, lit. on ground of, 'because of'). "Mechanically", the PP *in spite of the weather* consists of a P (in) and a complex NP (*spite of the weather*, itself containing a PP). *In spite of* here functions as a complex preposition, actually replaceable with the single preposition *despite*, making the syntactic structure of the PP simpler and more one-to-one with its semantics. Other examples of such LPs are phrases, often described using some kind of subcategorisation, are so called prepositional verbs *(tro på*, eng: believe in), V+NP+P

constructions seen as "nouns with supportive verbs" by Dura (1997) *(få grepp om*, lit. get grip about; 'understand') and complex auxiliary verbs *(kommer att*, 'will')[1].

My intuitive opinions as to what a lexicalised phrase is was initially shaped largely by an article by Anward & Linell (1976). Those opinions have since been somewhat altered but their article is still very important for my thinking on the subject of lexicalised phrases.

1.1 Criteria

I have two obligatory criteria that have to hold for every potential candidate as an LP. The first (1:a) is that the candidate should be a standard way of expression in Swedish; it should be an institutionalised way of saying something. The second (1:b) is the simple formal criterion that they must consist of more than one word but less than a clause.

In addition to these, either of the following two characteristics has to hold: (2:a) The candidate has to be semantically non-compositional OR (2:b) a "better" phrase-structure analysis is obtained when grouping the word combination in question as an LP.

Non-compositionality can either arise from the candidate being partially non-compositional, that is, one or more but not all words are used in a non-standard way, e.g. *fatta eld* (eng. 'catch fire'), where the word *fatta* is used in a marked way[2], or the candidate as a whole is used to mean something quite different than what is expressed with a literal interpretation[3] *(sticka under stol med,* 'hide').

Below are some examples of the reasoning behind criterion (2:b).

(1) [på [grund [av [faran]]]] lit: on ground of danger-DEF
(2) [på grund av [faran]]

A "mechanical" division of the above PP produces constituents whose semantic status is clearly debatable. Both *grund av faran* (ground of danger-DEF) and *av faran* (of danger-DEF) in (1) are uninterpretable without context[4]. I regard *på grund av* ('because of') as a complex preposition taking an NP, here *faran.* (See (2).)

(3) [Jag [tror [på [tomten]]]] lit: I believe on Santa-Claus-DEF
(4) [Jag [tror på [tomten]]]

In the VP *tror på tomten* (believes in Santa-Claus), the predicate meaning is described by the string *tror på* (believes in) and not by the verb (believes) alone. The PP *på tomten* in (3) is not meaningful without the verb. I see *tror på* as a complex transitive verb and the NP *tomten* here is the direct object. (See (4).)

(5) [Det [kommer [att regna]]] lit. It comes INF rain
(6) [Det [kommer att [regna]]]

Examples (5)-(6) treat the Swedish future tense expression. *Kommer att* is the string that expresses future tense in Swedish. *Kommer* alone does not, at least not in the written language. I see *kommer att* as a complex auxiliary verb.

Specific criteria are formulated for different structural types of lexicalised phrases where certain tests can be applied to candidates. Below are some examples:

Particle verbs

The general criterion saying that an LP should be semantically non-compositional should be applied to particle verbs in the following way:

Particle verbs that qualify as LPs should satisfy the following description: 1. The particle does not modify the verb meaning spatially in a way that can be predicted from the spatial meaning of the particle. *Slänga in* ('hurl in') could be an example of a non-qualifying candidate. However *lägga in* lit. 'lay in' qualifies because one of its meaning is non-compositional 'pickle'. 2. The particle verb is not an instance of a productive metaphor. Often spatial expressions can be used metaphorically as for instance: *slunga ut* 'hurl out' and *sväva ut* 'float away'. These do not qualify as LPs, because they are instances of productive spatial metaphors.

Reflexive verbs

The reflexive in a reflexive verb should not have argument status. The transitive reflexive verb *tvinga sig* 'force oneself' is thus not considered an LP.

Prepositional verbs

The noun phrase that follows the preposition should have argument status. It should also be replaceable with other noun phrases. If the whole of the verb+prep plus a following noun phrase forms an idiomatic phrase and the verb+prep combination can not be combined with other noun phrases, it is not a prepositional verb. The preposition plus the noun phrase should also not form an adverbial.

2. The use of "idioms" in tagging and parsing

The English Constraint Grammar, ENGCG (Karlsson et al 1995), uses a list of 700 syntactic "idioms" and 5000 complex nominals that are deterministically grouped and analysed as complex words at tokenisation. The two examples below are taken from Karlsson et al (1995).

in spite of --> in=spite=of/PP
run time --> run=time/NN NOM SG

The CLAWS 4 tagging system (Leech, Garside & Bryant 1994) used for annotating the British National Corpus (BNC) with parts-of-speech has a sub-component called IDIOMTAG, which makes use of multi-word units to correct prior erroneous taggings. The lexicon in IDIOMTAG has over 3000 entries representing "general idioms" (e.g. *as much as*), multi-word proper names (e.g. *Dodge City*) and foreign expressions (e.g. *annus horribilis*). IDIOMTAG introduces ambiguities where more than one "idiom" analysis is possible and later resolve these ambiguities. When Blackwell (1987) reported on an earlier version of CLAWS, IDIOMTAG made the idiom analyses deterministically, i.e. the

analyses could not be un-done at a later stage. From what I can make out of the documentation in Leech, Garside & Bryant (1994), the determinism is now abandoned, but I am not sure of that. The system can express several traits of variation and discontinuity of the "idioms".

Harald Berthelsen (Berthelsen 1997) used an earlier version of the LP lexicon that I will report on below and wrote a prolog program detecting LPs in texts. It was used to augment a Constraint Grammar (SWECG) analysis with LP membership on words. The analysed words had indexes pointing to the lexicon. Constraints could be written discarding or selecting LP analyses.

Other related implementations

Stephan Bopps (Bopp 1996) Phrase Manager is a framework for databases of multi-word units. The program makes it possible to specify e.g. transformation restrictions on idioms. Bopp does not report on an actual lexicon though. It is the database format itself that constitutes Phrase Manager.

André Schenk (Schenk 1994) analysed idioms and collocations and incorporated them into compositional M-grammar used for automatic translation in the Rosetta machine translation system. Schenk describes a formal apparatus for dealing with idioms and collocations but does not report on an actual lexicon either.

3. LP-DETECT

My system for detection of LPs (LP-DETECT) takes text files from the SUC Corpus (Ejerhed et al 1992) as input, converted and simplified so that only <word>:<tag> pairs remain. Optionally, sentence enumeration can also be displayed in the output. <word> are word forms except for verbs where <word> are base forms. <tag> are the POS tags from SUC. All other information such as the morphological features are deleted[5]. The output of the LP detection is also <word>:<tag> pairs but more information is added. 1. The words in the LPs are given an additional tag for LP membership. 2. The LPs are given an analysis. 3. The LPs are enclosed within square brackets.

Input: ... word:tag word:tag , ... , word:tag word:tag ...
Output: ... word:tag [word:tag_LPtag , ... , word:tag_LPtag LPanalysis] word:tag ...

LP-DETECT consists of three components, an LP lexicon, a detection program and Dis rules (**Dis**ambiguation and **Dis**carding).

3.1 The LP lexicon

The LP lexicon comprises 8057 LPs distributed over 4775 LP Sets. LP sets reflect the fact that many LPs have variants[6] and those variants are grouped together in the lexicon and are given the same main enumeration index. Examples of LP Sets are *vara/ligga/stå i maskopi med* ('be/lie/stand in collusion with') and *så in i baljan/norden/helvete* ('as hell'). The lexicon contains morpho-syntactic and some phonetic/prosodic information about the LPs. The phonetic/prosodic information is left from an earlier phase where LPs were collected to enhance text-to-speech synthesis (Lindberg 1996).

My main source for the LP lexicon has been Johannisson & Ljunggrens "Handordbok" (1966). To overcome problems connected with the age of "Handordbok", I have also used "Wörterbuch Der Schwedischen Phraseologie In Sachgruppen" (Schottman & Petersson 1989). I have also taken "slang" expressions from Haldo Gibson's "Svensk slangordbok" (1969). Other written sources have been "Svenska Akademiens ordlista över svenska språket" (1979) and Svenskt uttalslexikon (Hedelin et al 1989). Phrases following certain POS patterns or containing certain words have been excerpted from the SUC corpus (Ejerhed et al 1992) and the PAROLE corpus[7]. Collocations have also been excerpted from the SUC corpus and the Dragon corpus of Dept of Speech, Music and Hearing at KTH, Stockholm. I have also found a great deal of the phrases by leading an ordinary life; listening to people talk, watching TV and reading books, newspapers etcetera, always with a pencil and a paper within reach (if that counts as an ordinary life).

Table 1. The fields in LP entries with two examples.

Field name	**Example 1**	**Example 2**
Index number	*2076:2*	*4030*
Index word	*finna*	*hållet*
LP tag	*VT?VC?VI*	*AB*
Reg exp	*finna:VB_<RP>*	*helt:AB_och:KN_hållet:..*
Word info	....	3:obs!
Prosodic markers	1:acc;2:ob	1:acc;2:ob;3:acc
S-form indication	0	....
Imperative form indication	1	....
Phrase info	....	....
Internal POS structure	1:VB;2:PN	1:AB;2:KN;3:Y

The LP entries consist of ten fields and are examplified in Table 1 above. *Index number* is an enumeration of the LP sets. The second figure is an enumeration of entries within LP sets. *Index word* is an approximation of the least frequent word of the phrase, using the longest word of the phrase. The *LP tag* field contains a syntactic analysis for the LP as a whole. The tags for the *LP tag* field use the SUC part-of-speech analyses except for the verbal LPs where subcategorisation information is encoded. The LP in example 1 is given three possible analyses; as a mono-transitive, copulative or intransitive verbal LP. Example 2 is analysed as an adverbial LP. In the *Reg exp* field resides a regular expression used to find the LP in the input sentences. The regular expressions allow for alternatives both for words and tags as well as variables that are expanded in the main detection program described below. *Word info* is for various information such as the note *obs!* indicating that the word could have more than one pronunciation and only one is correct here. It can also contain information on tense agreement in LPs with more than one verb. The prosodic marker field contains information stating the extent to which the words of the LP retain their word accent when pronounced together. The next two fields bears information on allowed inflexional forms of the verb in verbal LPs. In the *Phrase info* field there may be additional morpho-syntactic information such as morphological features restricting the context of the LP. The last field is an enumeration of the parts-of-

speech in the phrase, following the SUC tagging conventions as strictly as possible. However, some words in LPs are notoriously difficult to annotate, since many LPs are grammatically deviant. Therefore two additional tags are used. One tag, X, is used for words that are only found in LPs, often archaic forms (e.g. *i **sinom tid,*** 'in due time'). The other tag, Y, is used for words that are used in a non-standard way in the LP (e.g. ***till sist,*** 'at last'). Also, the word tag NN (noun) has the morphological tags i=indefinite, d=definite, s=singular and p=plural as an extension of the SUC POS tag. In the current implementation, only the first four fields are used.

In Table 2 below, the phrase tags with subcategorisation codes for the verbal LPs are shown.

Table 2. Phrase tags with subcategorisation codes for the verbal LPs.

VI	Does not take a complement	VS	Takes a subordinate clause complement
VT	Takes an object (NP or S)	VTnexus	Takes an object (NP) LP-internally (in the nexus)
V2T	Takes two objects (NP or S)	VTnexusA	Takes an object (NP or S) LP-internally (in the nexus) and a post-posed *att*-clause as complements
Vadv	Takes an adverbial complement (of any form)	VTnexusS	Takes an object (NP or S) LP-internally (in the nexus) and a post-posed subordinate clause as complements
VH	Takes an infinitive VP	VTnexusH	Takes an object (NP or S) LP internally and a post-posed infinitive VP LP as complements
VC	Takes a predicative complement	V2Tnexus	Takes an object (NP, indirect object) LP-internally (in the nexus) and a post-posed direct object.
VA	Takes an *att*-clause (that-clause)	VHpass	Takes an infinitive VP with the verb in passive form

For certain verbal LPs, alternatives can be given:

VI?VC?Vadv *hålla_<RP>* Eng. lit: hold_REFL

Han var nödig men han höll sig (VI) 'He needed to go to the toilet but he restrained himself'
Hon höll sig underrättad (VC) 'She kept herself informed'
Han höll sig på kontoret hela dagen (Vadv) 'He stayed at the office all day'

The tags for the non-verbal LP analyses *(LP type)* are taken from the SUC part-of-speech tag set (where they of course are used to analyse words and not phrases). The one exception to that rule is the tag PD (multi-word pre-determiner). The LP analyses in question are shown in Table 3.

Table 3. Phrase tags for the non-verbal LPs with examples

AB	adverbial LP	*så länge*
DT	multi-word determiner	*en och annan*
NN	multi-word nominal	*mannen på gatan*
JJ	adjectival LP	*liten i maten*
PP	multi-word preposition	*tack vare*
SN	multi-word subordinating conjunction	*så länge*
KN	multi-word co-ordinating conjunction	*för att inte tala om*
IE	multi-word infinitive marker	*i akt och mening att*
PN	multi-word pronoun	*en och annan*
PD	pre-determiner	*en del av*
HP	multi-word interrogatory pronoun	*vem i all värden*
HA	multi-word interrogatory adverb	*hur i all världen*

3.2 The detection program

In the main detection program, written in perl, the regular expressions for the LPs are enhanced to allow for intervening material between certain words and specification of variables in LPs. Lookup speed is increased by only considering those LPs whose longest word is present in the sentence to be analysed (length of word approximating word frequency in an inverse relation). This is much more efficient than the usual method of using the first word as a trigger. When two or more words are of the same length, the last one is chosen for most structural types of LPs. The detection also allows for embedded LP analyses where one LP is embedded within another LP.

3.3 The Dis rules

The next step is the Dis rules. "Dis" stands for disambiguation and discarding of LPs. *Disambiguation*: The disambiguation rules pick one of several analyses for a given LP or discards an erroneous one, possibly leaving more than one analysis. *Discarding*: All LP analyses for a word sequence that is not an LP but happens to contain the words of an LP in the right order (juxtaposition) are deleted. Usually, word forms or tags in the immediate context are used for the Dis rules. 36 disambiguation rules and 75 discarding rules are implemented presently.

The algorithm for the detection of LPs with the Dis rules is shown below.

Algorithm:
Reads in and stores the LP lexicon
Regular expressions for LPs are augmented
For each SUC sentence:
    Converts the SUC file format
    Stores relevant lexicon entries
    For each stored lexicon entry matching the sentence:
        Marks matching words and gives the LP an analysis
    Runs dis rules if that is chosen and if at least one LP is found

4. Output

As indicated above, not all possible analyses residing in the lexicon are present in the output, partially because of the simplified input format. The examples (7)-(9) contain lexicon entries, output and an english traslation. Examples (8)-(9) also contain dis rules. The example entries only contain the information used in the implementation. In example (7), two LPs have been detected, the adverbial *i alla fall* ('at least') and a verbal LP taking a complement adverbial *ha det* (lit. have it).

(7)
*4236 fall AB **i**:PP_**alla**:DT_**fall**:NN*
*295:1 det Vadv (**få**:VB|**ha**:VB)_**det**:PN*

Men:KN östtyskarna:NN kan:VB nu:AB alltså:AB skryta:VB med:PP att:SN de:PN [**i**:PP_LP-4236 **alla**:DT_LP-4236 **fall**:NN_LP-4236 @AB-4236] [**har**:VB_LP-295:1 **det**:PN_LP-295:1 @Vadv-295:1] bättre:AB i:PP sängen:NN än:KN västtyskarna:NN .:MAD

Eng translation: But the east Germans can brag about the fact that they at least are better off in bed than the west Germans.

In example (8), a discarding rule has worked, deleting a particle verb reading (here VI, 'takes no complements') of the string *gå bort* (the idiomatic reading meaning 'die', here retaining the literal meaning 'walk away') using prepositions in the immediate right context.

(8)
Dis rule:
gå_PL^{8} *--> NIL / _ (mot|till|från)*

Gumman:NN [reste:VB_LP-2174 sig:PN_LP-2174 @VI-2174] och:KN [**gick**:VB_LP-2693 långsamt:AB **bort**:AB_LP-2693 @VI-2693] mot:PP sitt:PS bylte:NN .:MAD
>>>>>
Gumman:NN [reste:VB_LP-2174 sig:PN_LP-2174 @VI-2174] och:KN gick:VB långsamt:AB bort:AB mot:PP sitt:PS bylte:NN .:MAD

Eng translation: The old woman rose and walked away slowly towards her bundle.

In example (9), a disambiguation rule has chosen the analysis V2T (takes two object complements) using the right context (the LP *böna och be* means 'beg and plead'). Another disambiguation rule has chosen the analysis AB (adverbial LP) over SN (complex subordinating conjunction) for the LP *varje gång* 'each time', also using the right context.

54 böna V2T?VA?VI ***böna:VB_och:KN_be:VB***
*4771 gång AB?SN (****varje****:DT|****varenda****:DT)_****gång****:NN*

V2T?(VT?VA)?VI *--> V2T / __ (NN|PN) NPbörjan*
AB?SN *--> **AB** / __ (KN|PP|MID|MAD|VB)*

Hon:PN hade:VB [**bönat**:VB_LP-54 **och**:KN_LP-54 **bett**:VB_LP-54 @V2T?VA?VI-54] honom:PN att:IE vara:VB tystare:JJ ,:MID för:KN tänk:VB om:SN flickan:NN vaknade:VB och:KN [**varje**:DT_LP-4771 **gång**:NN_LP-4771 @AB?SN-4771] lovade:VB han:PN .:MAD
>>>>>
Hon:PN hade:VB [**bönat**:VB_LP-54 **och**:KN_LP-54 **bett**:VB_LP-54 @V2T-54] honom:PN att:IE vara:VB tystare:JJ ,:MID för:KN tänk:VB om:SN flickan:NN vaknade:VB och:KN [**varje**:DT_LP-4771 **gång**:NN_LP-4771 @AB-4771] lovade:VB han:PN .:MAD

Eng translation: She had begged him to be more quiet, because what if the girl would wake up, and each time he promised.

5. Small evaluation

I trained the system on 50.000 words of Swedish text from the SUC corpus. The genres represented newspaper text, legal text and novels. As a simple base-line setting test I tested the precision on one of the texts from novels from the training material. 144 sentences were scanned, 90 LPs were found. After the Dis rules 80 LPs remained. Below are the results. Figures in the evaluation are rounded because of the small amounts of text used.

90/144 = 0.6 LPs were detected per sentence as a mean
86/90 = 96 % of LPs were unambiguous initially, 100 % finally
76/90 = 84 % of LPs were correctly detected initially
75/80 = 94 % of LPs remaining after the dis rules were correctly detected

The precision figure rose from 84 % to 94 % using the dis rules. All ambiguities were eliminated.

To test recall, a text from a novel from the training material was manually scanned. It contained 144 sentences. 76 correct LPs had already been found by the detection program, nine more were found in the manual scan. Thus, the total number of LPs in the text was 85.

Recall: 76/85 = 89 %

Five of the misses were due to the LPs not being in the lexicon.

Lexicon recall: 80/85 = 94 %.

Since the lexicon lacked five LPs found in the manual scan, only 81 LPs would have been possible to find with the present lexicon, even with a perfect detection algorithm. 76 LPs were found. Therefore, the recall for the detection was 94 %.

Detection recall: 76/81 = 94 %

As for the shortcomings of the detection program, one flaw was due to the way the regular expressions work, scanning the texts from left to right. When a situation of LP overlap occurs, only the leftmost LP is detected. The other flaw was due to the very simple formulation on the allowed tags for intervening material in LPs as an unordered set of tags[9]. That forces prohibition of some tags, e.g. prepositions, conjunctions and verbs in order to avoid overgeneration. In fact, all tags can appear breaking up a verbal LP but not in any order. Other sources of misses were from faulty tagging in SUC.

I did a somewhat larger test in texts not from the training material. The proportions were 40 % newspaper text, 40 % text from novels and 20 % political text. 782 sentences were used. 415 LPs were found initially. After the Dis rules, 380 LPs remained.

415/782 = 0.53 LPs were detected per sentence as a mean
386/415 = 93 % unambiguous initially
369/380 = 97 % unambiguous finally
340/415 = 82 % correctly detected initially
339/380 = 89 % correctly detected finally

Here, the precision figure rose from 82 % to 89 % using the dis rules. That means that about a third of the initial false positives were eliminated by the dis rules. The proportion of unambiguous analyses rose from 93 % to 97 %. Thus more than half of the ambiguities were eliminated by the dis rules.

6. Discussion

My mini-tests have several flaws but I have chosen to present it here anyway just to get a rough idea of the magnitude of the problem of over- and undergeneration of the system. One flaw is the mere sparseness of the testing material. The different files varied considerably as to the number of false positives both before and after the discarding rules. Perhaps not surprisingly, there was a clear tendency for texts from novels to produce more false positives than those from the other two genres. The amount of remaining ambiguity in LP analyses was smaller for these texts though. There was also a tendency for newspaper texts to have more LPs in them but fewer different LPs, the use of LPs were more stereotyped in the newspaper texts.

The precision result on the small portion of the training material that was investigated shows that I have not been able to eliminate the overgeneration of LP analyses. Those cases that I could not write discarding rules for were either "parsning-complete" (i.e. a

parser not only for constituents but for grammatical functions would have been needed) or even "AI-complete" (i.e. only full text understanding would have sufficed to eliminate these false positives).

There were clear tendencies for the false positives. These instances were very often two-word verbal LPs ending in either a preposition (false prepositional verb), a particle (false particle verb) or a reflexive pronoun (false reflexive verb), where both the verb and the second word were high frequency words. Examples of false LPs are the prepositional verbs *vara till* 'have the function of', *bli till* 'become' and *gå för*[10]. The latter has later been reconsidered as not being a qualifying candidate as a prepositional verb by the author.

As for the recall figures, the method I used was not optimal. The author (also the compiler of the lexicon) scanned the text for LPs not found by the program and marked candidates that satisfied my criteria. A better way would have been to have another person look through the text and mark every LP that person could find using my criteria and after that run the program on the same material and compare the results.

Clearly, LP-DETECT deserves a better evaluation.

To sum up, the fact that about half of the sentences contained an LP as an average suggests that LPs are common enough to spend time on detecting in the first place. However, in addition to helping out a system of syntactic and/or semantic parsing, such a system could also be hindered by false positives, some of which are very difficult to avoid.

---

[1] Of course even idioms can be said to be valency breaking since syntactic arguments in idioms are often not part of the argument structure of the sentences where the idioms occur.

[2] Such phrases are often termed collocations (see e.g. Schenk 1994).

[3] A description most often attributed to idioms.

[4] Note that the Swedish genitive is not productively signalled by a PP with *av* (of) as in English. "The cause of the danger" is expressed *farans grund* (danger-DEF-GEN ground) in Swdish and not *grunden av faran* (ground-DEF of danger-DEF). The preposition *till* (to) can be used instead, however.

[5] Actually, the surface forms of the verbs are stored and retrieved later to be present in the output.

[6] See Sköldberg (1999) for an interesting discussion on types of idiom variation.

[7] The PAROLE corpus is part of "Språkbanken" at Dept of Swedish, Göteborgs University.

[8] **"gå_PL"** means "an LP consisting of some form of the verb *go* followed by a particle (PL)". Actually this is just a way of representing the Dis rules in a somewhat more readable form. In real life, the Dis rules are formulated as regular expressions.

[9] That is, <word>:<tag> pairs. The actual form of the regular expression for the intervening material is "([^:]+:(AB|PN|NN|DT|JJ|PC|PM|PS|RG|RO))*?".

[10] In the expression *det går för PERSON* ('PERSON is having a sexual climax') or in the expression *visa (PERSON1) vad PERSON2 går för* ('show PERSON1 what PERSON2 can do').

References

Anward, Jan & Linell, Per. 1976. Om lexikaliserade fraser i svenskan. *Nysvenska studier* 55-56, 77-119.

Berthelsen, Harald. 1997. The Constraint Grammar Idea applied to two Problems in Text-to-Speech: Detecting Multi-word Units and Prosodic Boundaries. D-level essay. Stockholm: Computational Linguistics, Stockholm University.

Blackwell, Susan. 1987. Syntax versus Orthography: Problems in the Automatic Parsing of Idioms (Ch 9). In: *The Computational Analysis of English. A Corpus-Based Approach.* (Eds. Roger Garside, Geoffrey Leech & Geoffrey Sampson) London & New York: Longman, 110-119.

Bopp, Stephan. 1996. *Phrase Manager: a System for the Construction and the Use of Multi-word Unit Databases.* EURALEX'96 Proceedings, 55-64.

Dura, Ela. 1997. *Substantiv med stödverb.* Meddelanden från Institutionen för Svenska Språket (MISS) 18. Göteborg: Göteborgs universitet..

Ejerhed, Eva; Källgren, Gunnel; Wennstedt, Ola & Åström, Magnus. 1992. *The Linguistic Annotation System of the Stockholm-Umeå Corpus Project, Version 4.31.* Umeå: Publications from the Department of General Linguistics, University of Umeå, no 32.

Gibson, Haldo. 1969. *Svensk slangordbok.* Stockholm: Bonniers.

Hedelin, Per; Jonsson, Anders & Lindblad, Per. 1989. *Svenskt uttalslexikon del 1-2.* Teknisk rapport nr 4. Inst för informationsteori. Göteborg: Chalmers University of Technology.

Johannisson, Ture & Ljunggren, K.G. 1966. *Svensk Handordbok. Konstruktioner och fraseologi.* Nacka: Svenska språknämnden och Esselte studium AB.

Karlsson, Fred; Voutilainen, Atro; Heikkilä, Juha och Anttila, Arto. (eds.) *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text.* Berlin - New York: Mouton de Gruyter.

Leech, Geoffrey; Garside, Roger & Bryant, Michael. 1994. *CLAWS4: The Tagging of the British National Corpus.* Kyoto: Proceedings of COLING Kyoto.

Lindberg, Janne. Detektering av lexikaliserade fraser för text-till-talkonvertering. 1996. *The Nordic Languages and Modern Linguistics.* Proceedings from the Ninth International Conference of Nordic and General Linguistics, (eds: Kjartan G. Ottósson, Ruth V. Fjeld and Arne Torp). Oslo: Novus, 191-203.

Målande uttryck. 1990. *En liten bok med svenska idiom.* Uppsala: Esselte ordbok, 1990.

Schenk, André Y. 1994. Idioms and Collocations in Compositional Grammar. Utrecht: OTS Dissertation Series.

Schottmann, Hans & Petersson, Ricke. 1989 *Wörterbuch Der Schwedischen Phraseologie In Sachgruppen.* Münstersche Beiträge Zur Deuchen Und Nordischen Philologie 6 1. Aufl. Münster: Kleinheinrich Verlag Für Kunst, Literatur Und Wissenschaft.

Sköldberg, Emma. 1999. *Varianter av idiom.* Svenskans beskrivning 23. (Eds. Lars-Gunnar Andersson, Aina Lundqvist, Kertin Norén & Lena Rogström). Lund: Lund University Press, 384-392.

Svenska Akademiens ordlista över svenska språket. 1979. Stockholm: P.A. Nordstedt & söners förlag.

Towards a Finite-State Parser for Swedish

Beáta Megyesi & Sara Rydin
Computational Linguistics
Department of Linguistics
Stockholm university
[bealsara]@ling.su.se

Abstract

In this study, we describe a method for parsing part-of-speech tagged unrestricted texts in Swedish using finite-state networks. We use the Xerox Finite-State Tool because of its expressiveness and power for writing and compiling regular expressions and relations. The parser is divided into four modules: i) contiguous phrase structure marker, ii) phrasal head marker, iii) syntactic function tagger, and iv) non-contiguous group boundary recognizer. The aim is to develop a parser that can be used as a light/shallow parser for marking phrase structure and, when needed, to label syntactic functions. We believe that modularity is necessary since different NLP tasks require various levels of analysis. The parser for Swedish is under development, but present-day results are promising.

1. Introduction

In several Natural Language Processing (NLP) tasks, such as information retrieval, information extraction, speech technology, machine translation, etc., full or partial information about phrasal and/or syntactic structures is needed. The main interest in these tasks lies in detecting the constituent structures and sometimes their syntactic functions in a robust and fast way. In this study, our aim is to develop a parser for Swedish part-of-speech tagged texts, based on finite-state techniques using the Xerox Finite-State Tool (Karttunen et al, 1997).

Finite-state techniques have been shown to be very useful for parsing unrestricted texts for several languages, such as English, Finnish, French, German, Swedish, etc. Under certain circumstances, these parsers are robust, fast and accurate. There are mainly three approaches that have been applied for the construction of finite-state parsers: constructive, reductionist, and the combinations of these.

Briefly, the *constructive* approach is based on lexical description of large collections of syntactic patterns using subcategorisation frames such as verbs and their arguments, and local grammars (Abney, 1996). The *reductionist* approach, on the other hand, starts from a large number of alternative analyses that get reduced through the application of constraints where the constraints may be expressed by a set of elimination rules (Voutilainen & Tapanainen, 1993) or by a set of restrictions applied in parallell (Koskenniemi et al, 1992). The *hybrid* method merges the constructive and the reductionist approaches. It is developed by Aït-Mokhtar and Chanod (1997) who built an incremental finite-state shallow parser for French in a modular way. The parser

makes incremental decisions throughout the parsing process. Syntactic information is added at the sentence level depending on the contextual information. They achieve broad coverage and include richer information than typical chunking systems.

A common procedure for building finite-state parsers from part-of-speech tagged texts is to first mark contiguous groups, e.g. noun or verb groups, then mark the heads within the groups and lastly, to extract patterns between non-contiguous group bounderies. However, Grefenstette (1996) points out that several parsers mix non-finite-state methods with finite-state procedures for different modules. He shows that the entire parser can be built easily within a finite-state framework by using finite-state transducers.

Finite-state transducers are finite-state machines that take an input and produce an output with each state transition. They generate or accept regular relations, i.e. sets of pairs of strings where each string has an upper and lower language. They can be written as regular expressions and can be used for introducing extra symbols into an input string, i.e. for labelling entities (groups) in a text. The labels, then, can be used for deriving further information from the text, such as extracting non-contiguous syntactic n-ary dependencies. By composing a sequence of transducers and dividing the parsing task into a sequence of partial tasks, such as contiguous group labelling, head marking, and the detection of non-contiguous group boundaries, Grefenstette presents a robust and fast light parser.

In this study, we use the Xerox finite-state tool (XFST), for constructing the parser. The reason for our choise is that the XFST is very convenient to use since it allows powerful and elegant linguistic descriptions by different operators for a high level of abstraction.

XFST is a general-purpose Unix application for computing with finite-state networks. Simple automata and transducers can be easily created by a set of operations from text files, binary files, regular expressions and other networks. Thus, XFST can read finite-state networks and compile them from regular expressions and text files. The networks can be simple finite-state automata or finite-state transducers and can be combined by various operations. In addition to the usual operators[1] (e.g. concatenation, union, optionality, Kleene star, Kleene plus, complement, intersection, relative complement, crossproduct, composition, etc.) XFST also supports some special operators for high level abstraction: restriction, replacement, and left to right longest match replacement. The *restriction* operator is very useful when writing constraints to exclude unwanted analyses. The rule A => B _ C expresses that A must appear in the context of B _ C, i.e. between B and C. The *replacement* operator replaces a string with another string with or without regard taken to context. For example, the rule A -> B|| L_R replaces A by B between a certain left and right context where A and B denote regular languages and the expression as a whole denotes a relation. The *longest match* operator is a special kind of replacement operator. It imposes a unique factorisation on every input. It can also be constrained by context and generalised for parallel replacement. For instance, the rule A @-> B ... C forces the transducer to locate and pick out maximal instances of the regular language A, leaving the entire string unchanged and inserting B and C around the selected A strings as markers.

2. The Parsing Method

The Swedish parser is based on the hybrid approach using a cascade of finite-state transducers. The parser consists, in its present form, of four modules: the phrase structure module, the phrasal head module, the syntactic function module and the non-contiguous group boundary module. The thought behind the modular architecture is to facilitate the work during development, to allow different uses of the parser and to reflect the different linguistic knowledge that is built into the parser.

First, modularity is important during the development of the parser as the modules (and the rules in each module) are ordered. Because information about the ordering of rules/modules and the separation of the linguistic knowledge are clearly specified, the detection of sources of incorrect analysis is facilitated. It is, because of the modularity, not only possible to use all four modules in the parser but to separate and run only the first module, the first and the second module, and so on. The first module would give us a phrase structure analyzed text and the sequential addition of the other three modules would introduce more and more information to the analysis.

Secondly, modularity can also be useful in regular use. There are, for example, NLP-tasks where only the information given by the first module is wanted; information about noun phrase and verb phrase boundaries can be used to identify events and entities in information extraction. There are also times when the syntactic function of the noun phrases is needed; information about the object can, in word sense disambiguation, be used to disambiguate the verb.

There is one last reason for modularity that is purely technical. With four modules, the whole set of rules is compiled into four separate finite-state networks. If all this information would be merged into one module, the compiled finite-state network would be quite large, the compilation would be time-consuming and the insertion of additional rules and the altering of the rule order would be more complex.

In the following, the finite-state networks describing the phrase and syntactic structure of the language are presented. As mentioned above, the parser consists of four networks, where each network is a composition of simple finite-state automata and/or finite-state transducers. Within each network, the transducers are composed and ordered in such a way that the easiest tasks are addressed first.

Each module marks up specific linguistic information by the use of reserved symbols, i.e. symbols that cannot be found in the natural language text files that are analyzed. The reserved symbols used by the parser consist of brackets and labels, shown in table 1 below.

Brackets	Label	Example	Comment
[/]	NP, PP, VP, AP	[NP	Left hand side phrase structure tag for NP
*	ActV, CopV, PasV InfV, HeadN, PrepN	*PrepN	Tag for head within a NP in a PP
{*** / ***}	Subj, Obj, Advl, PredF	{***Subj	Left hand side tag for subject
[/]	PVP	[PVP	Left hand side tag for particle verb phrase

Table 1. Symbols (tags) inserted by the parser

The corpus used to train and test the parser is the Stockholm-Umeå Corpus, so-called SUC (Ejerhed et al, 1992) annotated with the PAROLE tag set. A plus sign, a part-of-speech tag and an appropriate number of morphological tags follow each token in the text:

"...svenska+AQP0PN0S städer+NCUPN@IS..."

Before describing the modules that the parser consists of, the reader should be reminded that the parser is under development. This means that the description of Swedish in no way is fully correct or exhaustive. The goal has been to see how suitable XFST or finite-state networks are for building a parser for Swedish.

2.1 Phrase structure module

The first finite-state network module marks phrase structure for noun phrases (NP), verb phrases (VP), prepositional phrases (PP), adverb phrases (AdvP) and adjective phrases (AP). The parsing is done in a bottom-up fashion where the deepest constituents are analyzed first. Thus, adverb phrases are detected before the adjective phrases since AdvPs may be included in APs but not the other way around. In a similar way, APs are marked before NPs.

Example 1 below shows how the adjective phrase is marked up. First, an adverb phrase (ADVP) is defined as a word (Ord[4]), a part-of-speech tag "R" and a string of morphological tags (Tagg+). Second, the adjective phrase (AP) is defined as containing an optional adverb phrase, a word (Ord), the part-of-speech tag (A) followed by morphological tags (Tagg). Last, the regular expression for the insertion of the AP tag is defined with the help of the longest match and replacement operators.

```
define ADVP [Ord R Tagg+] ;
define AP [(ADVP " ") Ord A Tagg] ;
regex AP @-> "[AP " ... " AP]" ;
```

Example 1. Definition of and insertion of tags for the adjective phrase.

Next, the noun phrases are detected. Noun phrases are presently defined as being of three different types: 1. a single pronoun (PRON), 2. an optional determiner (DET) followed by one or more optional ordinal/cardinal numerals (NUM), followed by an optional adjective phrase (AP) (the last two can optionally be in reversed order), and at least one noun, 3. an optional determiner, a possessive pronoun (POSSPRON), an optional, tagged adjective phrase and at least one noun. The definition of NPs is given below.

```
define NP [[PRON]|[(DET) ([NUM]+) (AP) ([NUM]+) [NOUN]+]|
      [(DET) POSSPRON (AP) [NOUN]+]] ;
```

Only attributes that precede the noun are included in the NP. The regular expression for the insertion of tags for noun phrases (and VP and PP as well) is similar to the regular expression given for AP (see example 1). In the rest of this section, only the outlines of the definition rules are given in order to make the examples easier to understand.

Next, prepositional phrases are defined as consisting of either a preposition (PREP) followed by a noun phrase (NP), or of a composite preposition and a tagged noun phrase.

```
define PP [[[PREP]|[PREP KONJ PREP]] NP] ;
```

Last, the verb phrase can have two different forms depending on the sentence type the verb occurs in. First, the position of the verb in regular word order is defined as follows: an optional infinitive particle (INF, equivalent to the Eng. *to*), at least one verb (VERB) and an optional verb particle (PART). Secondly, the position of the verb is given in the case of subject-verb inversion: an auxiliary verb (AUX), followed by a tagged noun phrase (NP), followed by at least one verb (VERB) and an optional verb particle (PART).

```
define VP [[(INF ) [VERB]+ (PART)]|[AUX NP [VERB]+ (PART)]] ;
```

Below, example 2 shows text annotated with phrase tags where the phrase tags are in bold face. The sentence in English is: *"Fear of the disease forced the decision to build water mains and sewage pipes"*.

[NP Skräcken+NCUSN@DS **NP]** **[PP** för+SPS **[NP** sjukdomen+NCUSN@DS **NP] PP]** **[VP** tvingade+V@IIAS fram+QS **VP]** **[NP** beslut+NCNSN@IS **NP]** om+SPS **[VP** att+CIS bygga+V@N0AS **VP]** **[NP** vattenledningar+NCUPN@IS **NP]** och+CCS **[NP** avloppsrör+NCNPN@IS **NP]** .+FE

Example 2: Output from the Phrase Structure Module[2].

2.2 Phrasal head module

The output of the phrase structure module constitutes the input to the phrasal head module. Heads are marked in two types of phrases: verb phrases and noun phrases. The phrase head information is marked in order to be used later, in the definition of syntactic functions. Grefenstette (1996) suggested the division of noun and verb phrases into subcategories, though we decided on a different division[3] for the verb types. For noun phrases there are two tags; HeadN and PrepN. Tagging of noun phrase heads are done using two composed transducers. The first transducer tags all head nouns (i.e. the last noun or pronoun in the noun phrase) as HeadN while the second transducer alters the tag HeadN to PrepN when it occurs in a prepositional phrase. Note that this is not to say that the head of the PP is the noun (which would of course be wrong), but only a way to mark the noun in the PP and thereby differentiate NP included in PPs from the poor lonely ones.

The verb phrases have four types of head tags according to different subcategorisation frames: the active (ActV), the passive (PasV), the infinitive (InfV), and the copulative verb (CopV). The first three are easily defined and tagged on the basis of the morphological information given by the PAROLE tag set. Copulative verbs, on the other hand, are defined as one of the words 'bliva' ('become'), 'finnas' ('be', 'exist'), 'vara' ('be') and 'heta'/'kallas' ('be called'). Example 3 shows the output from the phrase head module.

> [NP ***HeadN** Skräcken+NCUSN@DS NP] [PP för+SPS [NP ***PrepN** sjukdomen+NCUSN@DS NP] PP] [VP ***ActV** tvingade+V@IIAS fram+QS VP] [NP ***HeadN** beslut+NCNSN@IS NP] om+SPS [VP att+CIS ***InfV** bygga+V@N0AS VP] [NP ***HeadN** vattenledningar+NCUPN@IS NP] och+CCS [NP ***HeadN** avloppsrör+NCNPN@IS NP] .+FE

Example 3: Phrasal head information inserted for the same sentence as in example 2.

2.3 Syntactic function module

In this module, an attempt is made to mark the syntactic functions of the phrases. We have elaborated with four kinds of syntactic functions: the subject, the object, the adverbial and the complement to the copulative verb. The annotation is done with help from the phrase tags and the head labels inserted by the previous modules. For example, NPs containing the HeadN label (in contrast to NPs with PrepN label) can be marked as subject or object. We have tried only to annotate syntactic function when fairly certain of the correctness of the result. Thus, we tried to avoid rules that would increase the recall but lower the precision substantially. Still, this module of the parser is probably the trickiest because of semantic and structural ambiguities.

Subject labeling is often dependent of both the left and the right context of the possible subject. There are several rules for the annotation of subjects and the choice among

them is done in the parser based on the word order in the sentence. Rules are primarily specified for regular word order (SV) and subject-verb inversion (VS). The scope of the subject is extended to include not only the noun phrase but also following adjacent prepositional phrases. This extension cannot be done for the object because of the PP-attachment ambiguity.

The annotation of objects is done on the basis of its left context. Here, the position of the verb and the already labeled subject are of interest since the object must follow the subject and/or the verb in a sentence. Note that the rule does not cover the case of topicalized objects, e.g. in the sentence 'Him she loved'.

The complement to the copulative verb is easily found since the copulative verb itself was annotated in the previous module by subcategorisation. Both object and verbal complements are expected to come after the verb but unfortunately, that is not always the case. Semantic features would be necessary to handle these phenomena correctly. Concerning adverbials only prepositional phrases are marked as such in the present module.

Lastly, an example of a sentence with syntactic function tags is given.

> **{***Subj** [NP *HeadN Skräcken+NCUSN@DS NP] **{***Advl** [PP för+SPS [NP *PrepN sjukdomen+NCUSN@DS NP] PP] **Advl***}** **Subj***}** [VP *ActV tvingade+V@IIAS fram+QS VP] **{***Obj** [NP *HeadN beslut+NCNSN@IS NP] **Obj***}** om+SPS [VP att+CIS *InfV bygga+V@N0AS VP] **{***Obj** [NP *HeadN vattenledningar+NCUPN@IS NP] och+CCS [NP *HeadN avloppsrör+NCNPN@IS NP] **Obj***}** .+FE
>
> *Example 4*: Output from the syntactic function module for the same sentence as in example 2 and 3.

2.4 Module for Non-Contiguous Group Boundaries

At the moment, this module incorporates only information about verb particles. Most of the particles are already found by the phrasal rules in the first module, i.e. when they follow directly after the verb. Here, those particles that are not adjacent to the verb are detected. The regular expression is quite straightforward as the verb particles have a separate tag in SUC and phrases between the verb and the particle are already marked up. In the future, we plan to incorporate other long distance dependencies, for instance in non-contiguous VP idioms.

3. Discussion

Presently, no extensive test or evaluation has been done on the parser since correctly labeled texts with phrase structure and syntactic information are not available. However, we tested the different modules on one text, consisting of 3000 words. The system accuracy regarding the detection of the different phrase structures seems to be good,

approximately 95%. The precision of marking syntactic features is lower approximately 60%-70%, because of syntactic ambiguity, such as PP-attachment, the scope of predicatives, complex NP structures and elliptic expressions. Recall is in both cases lower since our strategy has been to only label entities when fairly certain of the correctness of the result. As the reader realises, there is more work to do in order to develop a reliable parser.

However, we believe that the finite-state tool together with our parser architecture suits the requirements for a useful shallow parser. The advantage of our system is that it is fast, robust (in the case of the shallow parser) and modular. Because of the modularity, the user can choose between only analysing the phrase structure, that is the usual case, or adding even syntactic analyses when needed.

4. Conclusions

In this study, we presented a method for parsing part-of-speech tagged unrestricted texts in Swedish by using finite-state techniques in the Xerox Finite-State Tool. Because of the modular architecture of the parser, it can be used as a light/shallow parser for marking phrase structure and, when needed, to label syntactic functions. The different modules reflect different types of linguistic knowledge such as information on phrase structure, phrasal heads and syntactic functions. However, the parser for Swedish is under development. Due to the promising results we are planning to continue to improve upon the different modules.

Acknowledgements

We would like to thank the Department of Linguistics, Uppsala university, for giving us the opportunity to participate in the course 'Automata theory', and especially Torbjörn Lager who first introduced us to the XFST during this course.

Footnotes

[1] See Karttunen et al (1997) for a good description of the XFST operators.
[2] 'Ord' is defined as a string of accepted characters in the natural language that forms a word.
[3] Note that neither the maximal projection of the NPs ('vattenledningar och avloppsrör'), nor the PP consisting of a preposition and infinitive verb phrase ('om att bygga') are labeled in this module.
[4] Grefenstette (1996) parses verb and noun groups instead of phrases.

References

Abney, S. 1996. Partial Parsing via Finite-State Cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, 8-15 Prague, Czech Republic.

Aït-Mokhtar, S., & Chanod, J-P. 1997. Incremental Finite-State Parsing. In *Proceedings of ANLP'97*, 72-79, Washington.

Chanod, J. P. & Tapanainen, P. 1996. A robust finite-state grammar for French. *Techniqual report, Rank Xerox Research Centre*, Meylan, France.

Ejerhed, E., Källgren, G., Wennstedt, O. & Åström, M. 1992. The Linguistic Annotation System of the Stockholm-Umeå Corpus Project. *Report nr. 33,* Dept. of General Linguistics, University of Umeå.

Grefenstette, Gregory. 1996. Light Parsing as Finite-State Filtering. *Workshop on Extended Finite State Models of Language, ECAI-96*, Budapest, Hungary.

Karttunen, L., Chanod, J. P., Grefenstette, G., & Schiller, A. 1997. Regular Expressions for Language Engineering. *Natural Language Engineering* 2, 305--238, Cambridge University Press.

Koskenniemi, K. 1990. Finite-State Parsing and Disambiguation. In *Proceedings of the Thirteenth International Conference on Computational Linguistics COLING-90* 2, 229-- 232, Helsinki, Finland.

Koskenniemi, K., Tapanainen, P., & Voutilainen, A. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92* vol 1, 156-162, Nantes, France.

Voutilainen, A. & Tapanainen, P. 1993. Ambiguity resolution in a reductionistic parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, 394-403, Utrecht, Netherlands.

Semantic Clustering of Adjectives and Verbs Based on Syntactic Patterns

Costanza Navarretta
Center for Language Technology
Njalsgade 80 - Copenhagen
costanza@cst.ku.dk

Abstract

In this paper we show that some of the syntactic patterns in an NLP lexicon can be used to identify semantically "similar" adjectives and verbs. We define semantic similarity on the basis of parameters used in the literature to classify adjectives and verbs semantically. The semantic clusters obtained from the syntactic encodings in the lexicon are evaluated by comparing them with semantic groups in existing taxonomies. The relation between adjectival syntactic patterns and their meaning is particularly interesting, because it has not been explored in the literature as much as it is the case for the relation between verbal complements and arguments. The identification of semantic groups on the basis of the syntactic encodings in the considered NLP lexicon can also be extended to other word classes and, maybe, to other languages for which the same type of lexicon exists.

1 Introduction

The idea that the syntactic behaviour of words is connected with their meaning has been the assumption behind research in different fields such as lexical semantics and automatic clustering of words based on statistical methods. In particular much work has been done to describe the relation between the semantic characteristics of verbs and their syntactic patterns, among many Fillmore (1970) and Levin (1993), and to identify semantically similar words from large text corpora on the basis of their linguistic and distributional properties, i.a. Brown, della Pietra, de Souza, Lai & Mercer (1992), Pereira, Tishby & Lee (1993). Some research has also been done to extract the semantic meaning of adjectives on the basis of their co-occurrence with nouns, (Justeson & Katz 1993, Justeson & Katz 1995, Hatzivassiloglou & McKeown 1993, Hatzivassiloglou & McKeown 1997).

Justeson & Katz (1993) describe a method for disambiguating adjective senses by the nouns or the noun phrases they modify, using co-occurrences in large text corpora. They use statistical inference methods for organizing and analyzing the collected material. Their disambiguation method is based on the observation that certain nouns are strongly associated with some of the adjectives that modify them. For example the adjective *old* means "not-young" when combined with the noun "man",

but has the sense of "not-new" if occurring with the noun "house". Justeson and Katz disambiguate five common adjectives, *hard, old, light, right, short*, on the basis of their co-occurrence with sense-specific antonyms referring to opposite values of the same attribute (e.g. *old-new, old-young*).

Justeson & Katz (1995) investigate the semantic characteristics of the nouns which they used to disambiguate the five adjectives (Justeson & Katz 1993). Justeson and Katz find out that a few general semantic features such as *+/- animate, +/- concrete* are sufficient to characterize the disambiguating nouns. In the case of the adjective *hard* they also consider a syntactic construction in which the adjective does not modify a nominal, i.e. *it is hard/easy to do something.*

Hatzivassiloglou & McKeown (1993) describe a method for clustering adjectives semi-automatically according to their meaning in a parsed corpus as a first step towards the identification of adjectival scales. Their hypothesis is that adjectives describing the same property often modify the same set of nouns. The clustering method defined combines statistical techniques and linguistic information and relies on two similarity modules. Hatzivassiloglou and McKeown define *similarity* in terms of the distributional similarity of the adjectives in relation to the nouns they modify.

Hatzivassiloglou & McKeown (1997) identify constraints on the semantic orientation[1] of conjoined adjectives extracted from a large corpus. They combine statistical methods with morphological knowledge.

We follow the assumption that there is a connection between the syntactic behaviour and the meaning of words. Although we agree with Levin (1993) that "verb meaning is a key to verb behaviour", in this paper we go the other way round, i.e. from the syntactic behaviour of words we derive some of their semantic characteristics. In particular we have investigated to what extent it is possible to use the syntactic encodings of a corpus-based NLP lexicon to extract clusters of semantic related verbs and adjectives. Extracting semantic information from machine readable dictionaries has been the object of much research, i.a. (Vossen, Meijs & den Broeder 1989), (Wilks, Fass, Guo, McDonald, Plate & Slator 1989). Because we use an NLP lexicon, the data is already encoded in a structured way, making the extraction process straightforward. We have extracted adjectives and verbs sharing the same syntactic pattern in a corpus-based Danish NLP lexicon, the LE-PAROLE lexicon, and we have investigated to which extent the obtained clusters contained semantically "similar" elements. Because some syntactic constructions are common to a great number of adjectives and verbs, such as the simple attributive and/or predicative adjectival construction and the divalent verbal construction, these patterns cannot be used to cluster them. Instead we have extracted adjectives and verbs sharing more seldom patterns, such as adjectives subcategorizing for prepositional complements or taking expletives patterns.

Because the connection between verbal complements and verbal meaning has been widely studied, i.a. (Brent 1991, Levin 1993), the obtained clusters can be compared with semantic groups identified in the literature. Less studied is the connection between adjectival complementation and adjectival meaning[2].

In section 2 we give a definition of semantic similarity for adjectives and verbs, in 3 we

briefly introduce the LE-PAROLE Danish lexicon. In section 4 we present some examples of verbal semantic clusters extracted from the LE-PAROLE syntactic lexicon, while in 5 we give a few examples of the extracted clusters for adjectives. Finally in section 6 we propose a first evaluation of the obtained results and we make some concluding remarks.

2 A Definition of Semantic Similarity for Adjectives and Verbs

We define "similarity" of meaning for adjectives and verbs by parameters identified in the literature.

Adjectives have "similar" meaning if they are synonymous or antonymous (Miller, Beckwith, Fellbaum, Gross, Miller & Tengi 1993 (1990)) and if they belong to a linguistic scale (Hatzivassiloglou & McKeown 1993). Linguistic scales, according to the definition provided by Levinson (1983)[133], are "sets of linguistic alternates, or contrastive expressions of the same grammatical category, which can be arranged in a linear order by degree of informativeness or semantic strength". We relate adjectives belonging to the same scale, independently of their orientation, to a common "super-ordinate" concept.

Verbal linguistic scales exist, but they are not so frequent as adjectival scales, and only few verbs have "real" opposites. Thus we have extended the definition of similarity of verbs to include the *troponymy* relation. According to Miller et al. (1993 (1990)) verbs are troponyms if they are connected to a super-ordinate along more semantic dimensions. One of the most common relations holding among linguistic scales (Levinson 1983) and among many verbal troponyms (Miller et al. 1993 (1990)) is the entailment relation. In conclusion we consider verbs to be "similar" if they belong to a linguistic scale, are opposites, synonyms or troponyms.

3 The LE-PAROLE Lexicon

We have extracted adjectives and verbs using the syntactic encodings in the Danish LE-PAROLE lexicon which was produced in the EU-funded MLAP project LE-PAROLE. The Danish lexicon is one of 12 general language, monolingual electronic lexica for European languages encoded in SGML format according to a common model, the so-called PAROLE model[3]. This model guides the construction of generic NLP lexica, i.e. lexica which can be used in different applications and systems. The LE-PAROLE lexica are mainly encoded on the basis of the corpora collected by the LE-PAROLE corpus groups and the encodings in existing dictionaries.

The PAROLE model distinguishes three separate levels of description: morphology, syntax and semantics. At present the morphology and the syntax for 20,000 entries have been encoded[4]. A description of the PAROLE morphological and syn-

tactic levels can be found in (Guimier, Ogonowski & Partners 1998*a*) and (Guimier, Ogonowski & Partners. 1998*b*).

A simplified picture of the morphological and syntactic layers of the LE-PAROLE lexica can be seen in Figure 1.

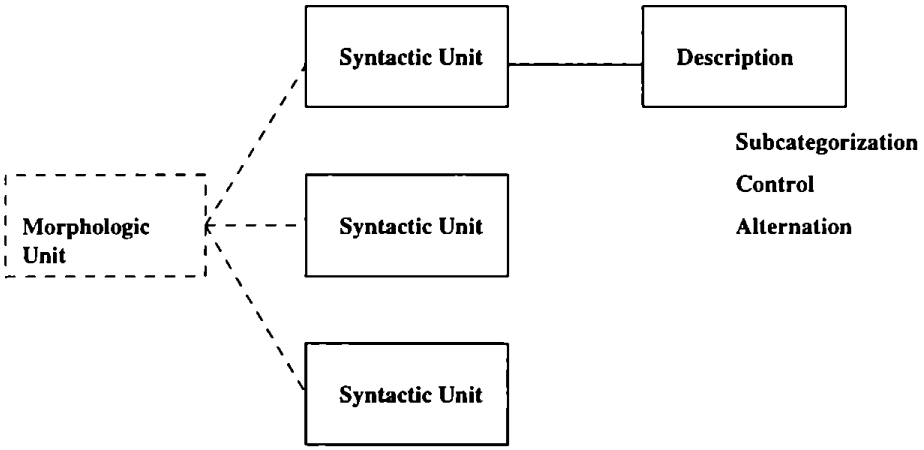


Figure 1: LE-PAROLE lexicon

The main entities of the morphological layer are Morphological Units (**MuS**) containing basic information on orthography, inflection and morphosyntactic features. One or more Syntactic Units (**SynUs**) are linked to each (**MuS**) and correspond to the syntactic patterns in which a morphological unit can occur. **SynUs** contain information about the syntactic behaviour of lexical units, such as sub-categorization, characteristics of the lexical unit when associated with a specific sub-categorization frame, control, diathesis alternations, linear order constraints. These information is encoded in the so-called **Description**. The Danish LE-PAROLE lexicon contains 20,000 morphological units. Of these units 2,816 are adjectival entries with their corresponding 3,304 syntactic units and 3,223 are verbal entries with corresponding 5,020 syntactic units.

4 Verbs

To verify the hypothesis that the syntactic encodings in an NLP lexicon can be used to extract semantically related verbs, we have looked at the syntactic patterns of verbs which belong to semantic groups recognized in the literature, in particular in (Levin 1993) and in WordNet (Miller et al. 1993 (1990)). Our study has shown that the elements of most of these groups share the same syntactic patterns (Description). Examples of verbal semantic clusters found by looking at the syntactic encodings in the Danish LE-PAROLE lexicon and the corresponding groups in other classifications are the following:

- Competition verbs (Miller et al. 1993 (1990)): *kæmpe* (battle), *fægte* (fence), *slås* (struggle), *stride* (fight), *konkurrere* (compete), *spille* (play) etc.
- Weather verbs (Miller et al. 1993 (1990)) (Levin 1993): *sne* (snow), *hagle* (hail), *regne* (rain), *blæse* (be windy) etc.
- Emotion verbs (Miller et al. 1993 (1990)) (Levin 1993): *genere* (bother), *pine* (torment), *fryde* (delight), "*bevæge*" ("move") etc.
- Verbs of Change of Possession (Levin 1993): *give* (give), *skænke* (donate), *forære* (present), *overdrage* (hand over), *testamentere* (leave by will) etc.

The verbs in each group share the same syntactic pattern, with the exception of the verbs of change of possessions which were obtained collecting verbs sharing three different descriptions. However, these descriptions are related and indicate the presence or absence of dative alternation and particular passive patterns where the second or the third complement (or both) can occur as subjects.

Emotion verbs share both a simple divalent pattern and a pattern with an expletive subject, an object and a clause as in the following examples:

Myggene generer mig
(The mosquitoes bother me)
Det generer mig at der er så mange myg
(It bothers me that there are so many mosquitoes)
Smerten piner hende
(The pain torments her)
Det piner mig at han ikke elsker mig mere
(It torments me that he does not love me any more)

Both patterns are also common to the motion verb *bevæge* used metaphorically as emotion verb:

Det bevægede ham at Maria havde husket hans fødselsdag
(It moved him that Maria had remembered his birthday)
Filmen bevægede ham dybt
(The film moved him deeply)

5 Adjectives

The patterns we have used to extract semantically related adjectives are predicative patterns where the adjectives subcategorize for prepositional phrases with nominal and clausal complements or raising constructions. The obtained groups have been checked manually and adjectives which were not semantically similar to the others have been removed. To validate the clusters we have also looked for corresponding synonyms and antonyms in WordNet. Finally we have identified common superordinates for each semantic cluster. In the following some of the obtained groups are given:

- "being afraid/not being afraid (in various degrees) of (doing) something": *bange* (afraid), *ræd* (scared), *angst* (fearful), *bekymret* (worried), *ubekymret* (carefree) ...
- "being easy/not easy (for somebody) to do something": *let* (easy), *nem* (simple), *besværlig* (troublesome), *vanskelig* (hard), *svær* (difficult)...
- "being irritated (in various degrees) at somebody": *gal* (mad), *vred* (angry), *sur* (irritated), *rasende* (raging), *forbitret* (furious)...
- "being happy/unhappy about something": *lykkelig* (happy), *ulykkelig* (unhappy)...
- "being friendly/not friendly (in various degrees) with somebody_1": *god* (kind), *sød* (nice), *venlig* (friendly), *flink* (nice), *streng* (strict), *styg* (nasty), *hård* (harsh), *modbydelig* (disgusting), *voldelig* (violent), *grusom* (cruel), *ond* (evil)...
- "being friendly/not friendly (in various degrees) with somebody_2": *god* (kind), *sød* (nice), *venlig* (friendly), *streng* (strict), *flink* (nice), *voldelig* (violent), *grusom* (cruel), *styg* (nasty), *modbydelig* (disgusting), *hård* (harsh), *ond* (evil) ...
- "being or not being capable (in various degrees) of doing something": *god* (good in the sense of capable), *snar* (quick), *fin* (good), *egnet* (fit), *flittig* (diligent), *flink* (good), *skrap* (sharp), *fortræffelig* (excellent), *enestående* (exceptional), *dygtig* (very good) *sød* (nice), *effektiv* (efficient), *langsom* (slow), *slem* (bad) ...

Although many of the elements in each group are also related by relations of synonymy, antonymy or hyponomy in WordNet, we have found more synonyms than in WordNet.

Some of the obtained groups had to be splitted up in more groups, such as the two groups "being angry (in various degrees) against somebody" and "being happy/unhappy about something" which share the same syntactic pattern. Some groups contained both related and unrelated adjectives. In the two groups "being friendly/not friendly (in various degrees) with somebody" the adjectives subcategorize for two different prepositions (*mod* (against) and *ved* (at)). We kept them separate because some Danes recognize a little semantic difference between the meaning of the adjectives in the two groups. It must be noted that the adjective *god* subcategorizing for the preposition *mod* can have two meanings depending on whether the prepositional nominal complement is animate or inanimate. In the former case the adjective belongs to the group we have identified, while in the latter case it means "effective" against something. Of course, we were not able to recognize this difference on the basis of the LE-PAROLE syntactic patterns.

6 Evaluation and Concluding Remarks

Before we evaluate the obtained results we must notice that the Danish LE-PAROLE lexicon only contains approximately 3,200 verbs and 2,800 adjectives and that only

some of the corresponding syntactic patterns have presently been encoded. The results obtained are based on this still incomplete lexicon. Although the Danish lexicon follows the common PAROLE model, the granularity chosen to identify syntactic patterns also depends on the lexicographic design chosen by the encoders. the results we have got also depend on these design choices.

our analysis of the extracted data has shown that all the groups of verbs and adjectives extracted from the LE-PAROLE Danish lexicon contain similar words. in the case of verbs all, or nearly all, the elements in the considered groups were semantically related. in few cases more "syntactic" groups formed a semantic cluster. the adjectival groups contained in some cases a few semantically unrelated elements besides the related ones and some of the adjectival syntactic groups had to be split up in different semantic clusters. The difference between verbal and adjectival behaviour is not surprising, because verbs have much richer, and thus more specialized, valency patterns than adjectives.

Although only unusual patterns, i.e. patterns which are shared by few words, can be used to identify semantically related words, and although the groups must be manually checked, we believe that the obtained results are quite interesting especially for adjectives, where the relation between syntactic pattern and meaning has not been exploited as much as it is the case for verbs. Another positive result is that we found more synonyms and antonyms than in WordNet for both verbs and adjectives.

In our opinion, semantic classifications of words must combine top-down with bottom-up strategies. Clustering words on the basis of their distributional behaviour in large corpora or their syntactic patterns in NLP corpus-based lexica is a valuable way to complement the top-down classification process. We believe also that the results obtained in our study, show that lexica with rich and well defined information as the lexica which follow the PAROLE model can be used to identify semantical related clusters and help in exploiting regularities/irregularities in the use of language.

Future work consists in extracting more groups of adjectives and verbs from the LE-PAROLE lexicon and analyzing them.The study should also be extended to complement-taking nouns and to adverbs. Because LE-PAROLE lexica, and/or NLP lexica containing the same type of syntactic information as these, exist for other European languages, the correspondence between syntactic behaviour and semantic meaning in more languages can also be investigated. The standardized encodings of the LE-PAROLE lexica offer new possibilities of analyzing alternations and other phenomena and of comparing them across different languages.

Footnotes

[1]Semantic orientation is also called polarity in the literature.

[2]Most of the proposed taxonomies for adjectives are not related to their syntactic behaviour. An exception is the taxonomy proposed in (Vendler 1963). A review of existing studies on the meaning of adjectives can be found in (Raskin & Nirenburg 1995).

[3]For a general description of the PAROLE model the reader is referred to (Calzolari 1996).

[4]The on-going European-funded project SIMPLE is in charge of encoding part of the semantic

level, i.a. (Pedersen & Keson 1999). The Danish STO project (Braasch, Christensen, Olsen & Pedersen 1998) is extending the vocabulary of the Danish LE-PAROLE lexicon to cover domain-specific words. However in this paper we exclusively work with the syntactic encodings in the LE-PAROLE lexicon.

References

Braasch, A., Christensen, A. B., Olsen, S. & Pedersen, B. (1998). A Large-scale Lexicon for Danish in the Information Society. *Proceedings from the First International Conference on Language Resources and Evaluation*, Granada, pp. 249–254.

Brent, M. (1991). Semantic Classification of Verbs from their Syntactic Contexts: An Implemented Case Study of Stativity. *Proceedings of the 5th European ACL Conference*, pp. 222–226.

Brown, P. F., della Pietra, V. J., de Souza, P. V., Lai, J. C. & Mercer, R. L. (1992). Class-based N-gram Models of Natural Language. *Computational Linguistics* **18**(4), 467–479.

Calzolari, N. (1996). PAROLE Linguistic Resources: Technical Specifications Overview. MLAP PAROLE 4, Pisa: CNR.

Fillmore, C. J. (1970). The grammar of *Hitting* and *Breaking*. R. Jacobs & P. Rosenbaum, eds. *Readings in English Transformational Grammar*. Ginn, Waltham, MA.

Guimier, E., Ogonowski, A. & Partners, P. (1998*a*). Report on the Morphological Layer. LE-PAROLE Report P-WP1.1-MEMO-ERLI-32, ELRI.

Guimier, E., Ogonowski, A. & Partners., P. (1998*b*). Report on the Syntactic Layer. LE-PAROLE Report P-WP1.1-MEMO-ERLI-33, ELRI.

Hatzivassiloglou, V. & McKeown, K. (1993). Towards the Automatic Identification of Adjectival Scales: Clustering Adjectives according to their Meaning. *ACL Proceeding, 31st Conference*, pp. 172–182. Columbus, Ohio, USA.

Hatzivassiloglou, V. & McKeown, K. (1997). Predicting the Semantic Orientation of Adjectives. *EACL Proceeding, 8th Conference*, pp. 174–181. Madrid, Spain.

Justeson, J. & Katz, S. (1993). Principled Disambiguation: Discriminating Adjective Senses with Modified Nouns. *Making Sense of Words, Proceedings of the 9th Conference of the UW Centre for the New OED and Text Research*, pp. 57–73. Oxford England.

Justeson, J. & Katz, S. (1995). Principled Disambiguation: Discriminating Adjective Senses with Modified Nouns. *Computational Linguistics* **21**(1), 1–27.

Levin, B. (1993). *English Verb Classes and Alternations*. Chicago: The University of Chicago Press.

Levinson, S. C. (1983). *Pragmatics.* Cambridge, England: Cambridge University Press.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. J. & Tengi, R. (1993 (1990)). 5 Papers on Wordnet. CSL report 43, Cognitive Science Laboratory, Princeton University.

Pedersen, B.S. & Britt Keson (1999). 'SIMPLE - Semantic Information for Multifunctional Plurilingual Lexica: Some Danish Examples on Concrete Nouns'. *SIGLEX99: Standardizing Lexical Resources, Association of Computational Linguistics.* ACL99 Workshop, Maryland.

Pereira, F., Tishby, N. & Lee, L. (1993). Distributional Clustering of English Words. *Proceedings of the 31st Annual Meeting of the ACL.* pp. 183–190. Columbus, Ohio.

Raskin, V. & Nirenburg, S. (1995). *Lexical semantics of adjectives.* Technical Report MCCS-95-288, Computing Research Laboratory - New Mexico State University.

Vendler, Z. (1963). 'The Grammar of Goodness'. *The Philosophical Review*, pp. 446–465.

Vossen, P., Meijs, W. & den Broeder, M. (1989). *Computational Lexicography for Natural Language Processing.* chapter: 'Meaning and Structure in Dictionary Definitions', pp. 171–192. UK: Longman.

Wilks, Y., Fass, D., Guo, C.-M., McDonald, J., Plate, T. & Slator, B. (1989). *Computational Lexicography for Natural Language Processing.* chapter: 'A Tractable Machine Dictionary as a Resource for Computational Semantics', pp. 193–228. UK: Longman.

An HPSG Account of Danish Pre-nominals

Anne Neville
Center for Language Technology
Copenhagen University
anne@cst.ku.dk

Abstract

This article addresses the issue of selection restrictions for noun phrase specifiers. Danish data is presented which shows that definiteness plays an important role in this respect. It is pointed out that an analysis is required in which the specifier, when present, leaves a mark on the projected phrase. This is achieved by assuming that specifiers are syntactic heads of noun phrase constructions. Further an elaborate classification of specifiers is also needed in terms of which selection restrictions may be formulated, along with a cross-categorial definiteness feature. These properties are part of the analysis proposed in this analysis.

Introduction

When investigating empirical data it becomes clear that noun phrases often have multiple specifiers appearing before the noun. An important goal of noun phrase analysis is the specification of selection restrictions for noun phrase specifiers and pre-nominals in general to account for combinations of specifiers. It is this goal that is pursued in this article.

In section 1 a set of Danish noun phrases are presented which form the basis of a discussion of what properties determine the restrictions on combinations of pre-nominals. In section 2 a number of previous HPSG analyses of noun phrases and pre-nominals are discussed. In section 3 the proposed analysis is introduced and sample analyses are shown. The proposed analysis has been implemented in the LKB system (Copestake 1999). A test suite consisting of the data in section 1 has been run and the results are presented in section 4. The article is concluded in section 5.

1 Some data

To narrow the focus of this article two sets of noun phrases have been selected which illustrate the importance of the feature of definiteness in the account of selection restrictions. One set consists of noun phrases with multiple definite specifiers,

as shown in (1), and the other set contains pre-nominals showing the dependence between specifiers and adjectives wrt. definiteness, as shown in (2).

(1) a. denne min begejstring
(this my enthusiasm)

b. denne den sidste Rabbit-bog
(this the last Rabbit book)

c. jeres den gamle grammofon
(your the old record player)

d. deres den nærmeste nabo
(their the nearest neighbour)

e. hende den tossede malerinde
(her the crazy painter)

f. ham den forsvundne dreng
(him the missing boy)

(2) a. de tre mindste skoler
(the three smallest schools)

b. * tre mindste skoler
(three smallest schools)

c. de mange smukke ting
(the many beautiful things)

d. * de mange smukkeste ting
(the many most beautiful things)

All the examples in (1) contain two definite specifiers. In (1a) the demonstrative specifier *denne* precedes a possessive specifier. In (1b) the demonstrative specifier *denne* precedes the definite article. In (1c) and (1d) the possessive specifiers *jeres* and *deres* likewise precede the definite article. Finally, in (1e) and (1f) the personal pronoun specifiers *hende* and *ham* precede the definite article[1]. To account for these combinations, ruling out all other combinations, a detailed classification of definite specifiers is required. In (2) definiteness determines the possibility of combining specifiers and adjectives. In (2a) the specifier *tre* (three) combines with the definite adjective *mindste* (smallest). However, the unacceptability of (2b) indicates that this combination is licensed by the presence of the definite article. (2c) and (2d) show that *mange* (many) does not allow a following definite adjective, irrespective of the presence of a preceding definite article. Thus definiteness is not confined to the description of specifiers, but it is also relevant to the analysis of adjectives.

2 Previous analyses

A number of analyses of pre-nominals have been proposed within the framework of HPSG. Pollard & Sag (1994) propose a noun phrase analysis which is an NP

analysis for English. Pre-nominals are divided into specifiers and modifiers, and one specifier is allowed in an NP. The specification of selection restrictions consequently does not become relevant[2]. A series of alternative analyses have been put forward since (Netter 1994, Kolliakou 1995, Allegranza 1998, Kathol 1998 and Börjars 1994).

Netter (1994) proposes a "DP" analysis for German. DP is in quotes because the analysis is not a DP analysis in the traditional sense (Abney 1987 and Delsing 1993). It is a DP analysis in the sense that the determiner is the head. However, determiners and nouns are assumed to be subtypes of a common nominal supertype. This means that both a noun phrase with a determiner and a noun phrase without a determiner may function as a maximal nominal phrase. Netter's analysis allows for multiple specifiers, in theory. However, he makes no attempt to specify selection restrictions.

Kolliakou (1995) also proposes a "DP" analysis, for Greek. It is a DP analysis in the same sense as Netter's analysis. However, Kolliakou's nominal type hierarchy is much more detailed than Netter's, and in addition to determiner and noun types she proposes demonstrative, numeral and adjective, all as subtypes of a common nominal supertype. Her analysis allows for multiple specifiers, and she specifies selection restrictions for them in terms of the nominal type hierarchy. Significantly, her analysis covers quantifying specifiers. Kolliakou's analysis covers a wide range of noun phrases. Her hierarchy is developed for Greek, and does not account for the Danish data, though.

Allegranza (1998) puts forward an "NP" analysis for Italian. Here NP is in quotes because it is not an NP analysis in the traditional sense (Chomsky 1970) and Jackendoff 1977). The noun is the syntactic head, but he introduces a marking feature by way of which the specifier non-head leaves a mark on the projected noun phrase. The value of the marking feature is a nominal type hierarchy as in the above-mentioned analyses. In his analysis determiner is a common supertype of a number of nominal subtypes. Allegranza's analysis likewise accounts for multiple specifiers, and selection restrictions are based on the determiner type hierarchy. His analysis also covers quantifying specifiers. Like Kolliakou's analysis, Allegranza's analysis covers a wide range of noun phrases. His analysis is developed for Italian, and again does not account for the Danish data without modifications.

Kathol (1998) proposes another "DP" analysis for English. Kathol also bases his analysis on a nominal type hierarchy, where determiner and nouns are subtypes of a common type. But like Pollard and Sag's analysis, his analysis only allows for one specifier, which makes it unable to account for multiple specifiers.

Finally, Börjars (1994) proposes an analysis very similar to Pollard and Sag's account. Consequently it has the same drawbacks as their account. However, it is interesting because it introduces definiteness as a syntactic primitive. It has already been shown how definiteness plays an important role in specifying selection restrictions. Börjars does, however, not explore the full potential of the feature.

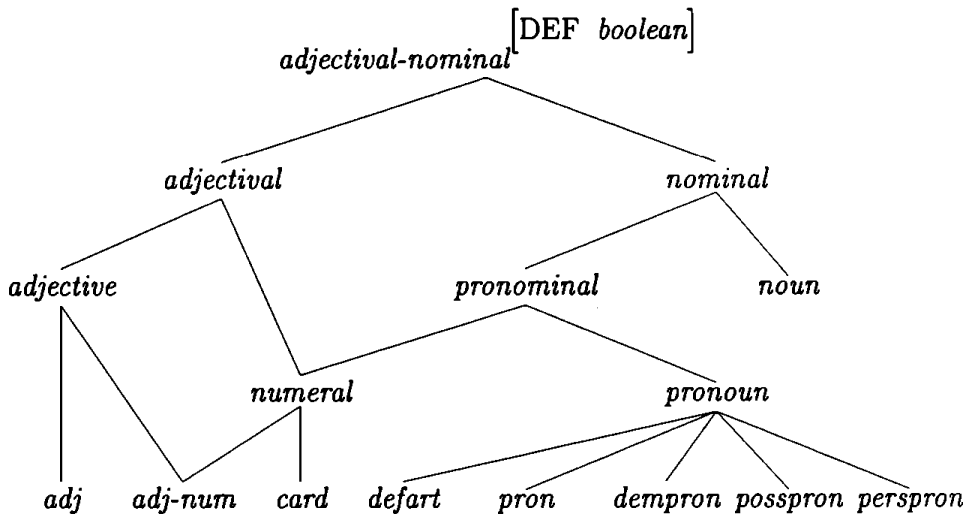
The analyses referred to here serve to show that two properties of noun phrase analysis are important. Firstly, an analysis is required in which the specifier, when present, leaves a mark on the projected phrase. This can be achieved either by assuming that the specifier is the syntactic head, or by introducing a marking feature by way of which the specifier marks the projected noun phrase. Secondly, an elabo-

rate classification of specifiers is needed in terms of which selection restrictions may be formulated. Börjars' account further supports the observation that definiteness is a key feature in the analysis of noun phrases.

3 Proposed analysis

The analysis here builds on ideas from the accounts presented in section 2. A division into modifier and specifier pre-nominals is assumed where specifiers are analyzed as heads selecting their non-head sister. It is based on a subtyping of the HPSG *head* type into a hierarchy of adjectival and nominal types. It sets itself apart in a number of ways. First and foremost the analysis is distinguished by its emphasis on definiteness. This is reflected by the detail with which definite pronouns are subtyped, and by the adoption of a separate feature of definiteness pertaining to all adjectival and nominal categories. The subtyping of *adjectival-nominal* is shown in (3).

(3)



The attribute DEF is defined for the *adjectival-nominal* type, which means that it is inherited by all the subtypes of *adjectival-nominal*. The type hierarchy and the assumption that specifiers are syntactic heads of noun phrase constructions enable an account of the data in section 1. In the following two sample analyses are presented. The sample noun phrases are *hende den tossede malerinde* (her the crazy painter) and *tre mindste skoler* (three smallest schools).

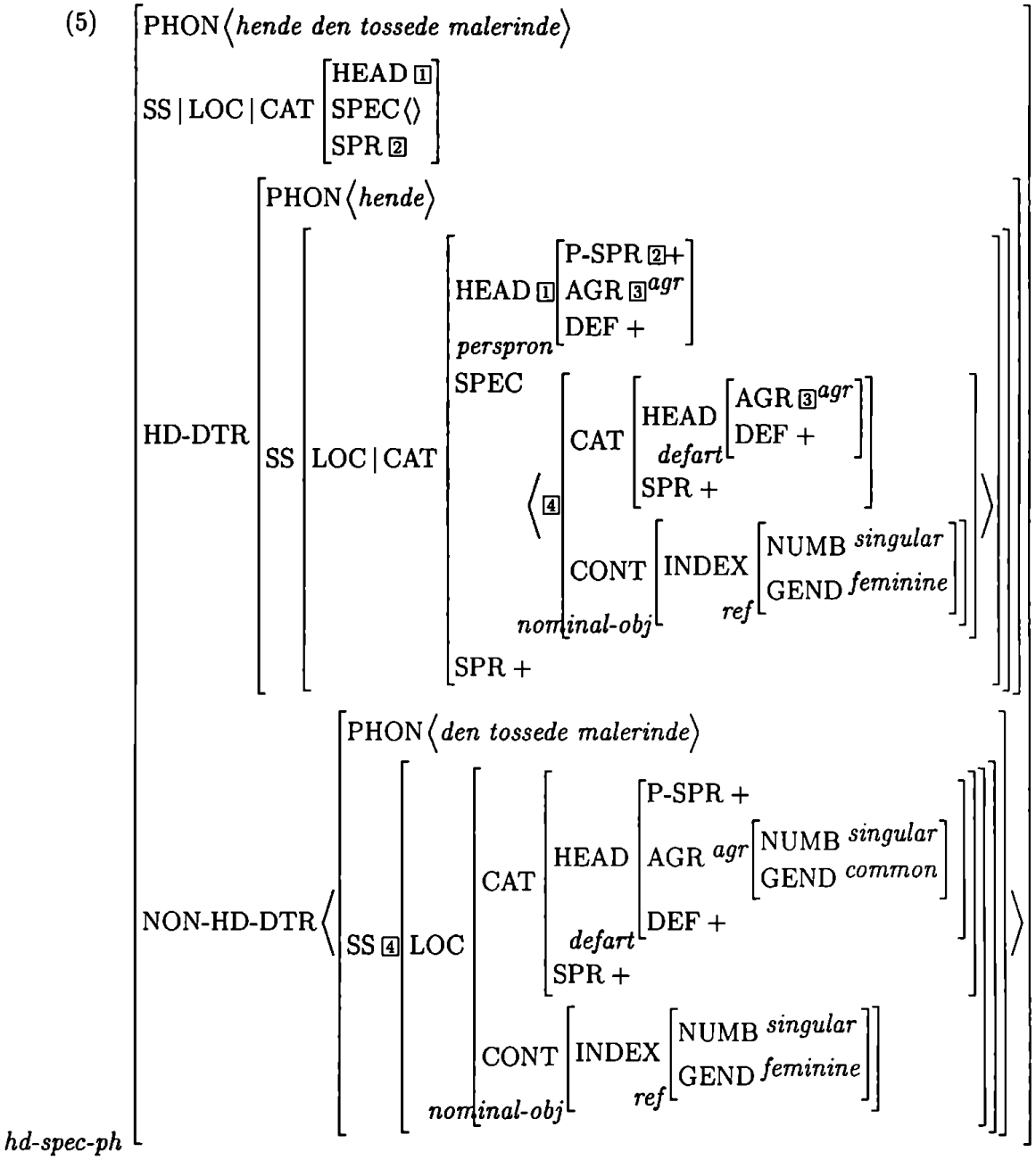
In (4) the lexical entry and thus selection restriction for the personal pronoun specifier *hende* (her) is shown.

(4)
$$\left[\begin{array}{l}\text{PHON}\langle \textit{hende}\rangle \\ \text{SS}\,|\,\text{LOC}\,|\,\text{CAT}\left[\begin{array}{l}\text{HEAD}\ \textit{perspron}\left[\begin{array}{l}\text{P-SPR}+\\ \text{AGR}\,\boxed{1}^{agr}\\ \text{DEF}+\end{array}\right] \\ \text{SPEC}\left\langle\left[\begin{array}{l}\text{CAT}\left[\begin{array}{l}\text{HEAD}\ \textit{nominal}\wedge\neg\textit{perspron}\left[\begin{array}{l}\text{AGR}\,\boxed{1}^{agr}\\ \text{DEF}+\end{array}\right]\\ \text{SPR}+\end{array}\right]\\ \text{CONT}\ \textit{nominal-obj}\left[\begin{array}{l}\text{INDEX}\ \textit{ref}\left[\begin{array}{l}\text{NUMB}\ \textit{singular}\\ \text{GEND}\ \textit{feminine}\end{array}\right]\end{array}\right]\end{array}\right]\right\rangle \\ \text{SPR}\,\boxed{2}\end{array}\right]\end{array}\right]$$

The value of HEAD in (4) shows that *hende* is of type *perspron* and DEF+. The P-SPR attribute indicates whether *hende* projects a maximal noun phrase or not. (4) further shows that *hende* is a specifier, the value of the SPEC attribute is a list restricting its non-head selectee. The selection restriction is that the selectee must be a *nominal*, but not a *perspron*. In addition it must be DEF+ and SPR+, i.e. already maximal..

In (5) the analysis of the noun phrase *hende den tossede malerinde* (her the crazy painter) is given.

(5)



(5) shows that the noun phrase is indeed a maximal well-formed noun phrase. The selection restriction has been resolved by the unification of the underspecified selection constraint and the actual occuring constituent headed by a definite article, *defart*. The value of the specifier head's P-SPR attribute is structure shared with the SPR value of the projected phrase, the latter indicating whether the phrase is a maximal noun phrase, which it is in this case as it is SPR+.

In (6) and (7) the lexical entries for the cardinal specifier *tre* (three) are shown.

(6)
$$\left[\begin{array}{l}\text{PHON}\langle \textit{tre}\rangle \\ \text{SS}\,|\,\text{LOC}\,|\,\text{CAT}\left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{l}\text{P-SPR}+ \\ \text{AGR}\,\boxed{1}^{agr}\left[\text{NUMB}\ \textit{plural}\right] \\ \text{DEF}- \end{array}\right] \\ \text{SPEC} & \left\langle \left[\begin{array}{ll}\text{CAT} & \left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{l}\text{AGR}\,\boxed{1} \\ \text{DECL}\ \textit{strong} \\ \text{DEF}- \end{array}\right] \\ & \neg\textit{pronominal} \\ \text{SPR}- & \end{array}\right] \\ \text{CONT} & \textit{nominal-obj}\end{array}\right]\right\rangle \\ \text{SPR}\,\boxed{2} & \end{array}\right]\end{array}\right]$$

(7)
$$\left[\begin{array}{l}\text{PHON}\langle \textit{tre}\rangle \\ \text{SS}\,|\,\text{LOC}\,|\,\text{CAT}\left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{l}\text{P-SPR}- \\ \text{AGR}\,\boxed{1}^{agr}\left[\text{NUMB}\ \textit{plural}\right] \\ \text{DECL}\,\boxed{2} \\ \text{DEF}\,\boxed{3} \end{array}\right] \\ \text{SPEC} & \left\langle \left[\begin{array}{ll}\text{CAT} & \left[\begin{array}{ll}\text{HEAD} & \left[\begin{array}{l}\text{AGR}\,\boxed{1} \\ \text{DECL}\,\boxed{2} \\ \text{DEF}\,\boxed{3} \end{array}\right] \\ & \neg\textit{pronominal} \\ \text{SPR}- & \end{array}\right] \\ \text{CONT} & \textit{nominal-obj}\end{array}\right]\right\rangle \\ \text{SPR}\,\boxed{4} & \end{array}\right]\end{array}\right]$$

There are two entries for *tre* (three), and only (6) projects a maximal noun phrase because it is P-SPR+. The selection restriction for this version is something which is not *pronominal* and DEF−. The other version, (7), does not project a maximal phrase. The constraint on its selectee is also something which is not *pronominal*, however, there is no constraint on the value of DEF.

In (8) the analysis of *tre mindste skoler* (three smallest schools) is shown.

(8)
$$\left[\begin{array}{l}\text{PHON}\langle \textit{tre mindste skoler}\rangle \\ \text{SS}\,|\,\text{LOC}\,|\,\text{CAT}\left[\begin{array}{l}\text{HEAD}\,\boxed{1} \\ \text{SPEC}\langle\rangle \\ \text{SPR}\,\boxed{2}\end{array}\right] \\ \text{HD-DTR}\left[\begin{array}{l}\text{PHON}\langle \textit{tre}\rangle \\ \text{SS}\left[\begin{array}{l}\text{LOC}\,|\,\text{CAT}\left[\begin{array}{l}\text{HEAD}\,\boxed{1}\left[\begin{array}{l}\text{P-SPR}\,\boxed{2}- \\ \text{AGR}\,\boxed{3}\left[\begin{array}{l}\text{NUMB}\ \textit{plural}\end{array}\right]\end{array}\right]_{\textit{agr}} \\ \text{SPEC}\left\langle \boxed{4}\left[\begin{array}{l}\text{HEAD}\left[\begin{array}{l}\text{AGR}\,\boxed{3} \\ \text{DEF}\,+\end{array}\right]_{\textit{adj}} \\ \text{SPR}\,-\end{array}\right]\right\rangle \\ \text{SPR}\,-\end{array}\right]_{\textit{card}}\end{array}\right]\end{array}\right] \\ \text{NON-HD-DTR}\left\langle\left[\begin{array}{l}\text{PHON}\langle \textit{mindste skoler}\rangle \\ \text{SS}\,\boxed{4}\left[\begin{array}{l}\text{LOC}\,|\,\text{CAT}\left[\begin{array}{l}\text{HEAD}\left[\begin{array}{l}\text{P-SPR}\,- \\ \text{AGR}\ \textit{agr}\left[\begin{array}{l}\text{NUMB}\ \textit{plural} \\ \text{GEND}\ \textit{common}\end{array}\right] \\ \text{DEF}\,+\end{array}\right]_{\textit{adj}} \\ \text{SPR}\,-\end{array}\right]\end{array}\right]\end{array}\right]\right\rangle\end{array}\right]_{\textit{hd-spec-ph}}$$

(8) shows that this phrase is not a maximal well-formed noun phrase. The selection restriction has been resolved by the unification of the underspecified selection constraint and the non-head constituent headed by an adjective, i.e. *adj*. The value of the specifier head's P-SPR attribute is again structure shared with the SPR value of the projected phrase. The projection is SPR– and consequently not a maximal noun phrase. What is important to note is that the version of the cardinal which projects maximal phrases cannot be used here as it constrains its selected constituent to be DEF–, but *mindste skoler* (smallest schools) is headed by a DEF+ adjective.

The two analyses show how specifier heads, the hierarchy and the definiteness feature are exploited to achieve correct analyses of the data.

4 Test with LKB

The proposed analysis of Danish noun phrases has been implemented in the LKB system which is a grammar and lexicon development environment for use with constraint-based formalisms (Copestake 1199). The system has been extensively tested with grammars based on HPSG theory. To test the analysis, a test suite

consisting of the data in section 1 has been parsed with the implemented grammar. The results are shown in (9)

(9) 1 denne min begejstring 1
2 denne den sidste Rabbitbog 1
3 jeres den gamle grammofon 1
4 deres den nærmeste nabo 1
5 hende den tossede malerinde 2
6 ham den forsvundne dreng 2
7 de tre mindste skoler 2
8 * tre mindste skoler 0
9 de mange smukke ting 2
10 *de mange smukkeste ting 0
;;; Total CPU time: 1240 msecs

The figure after each phrase gives the number of parses found. Examples 5, 6 and 9 get two parses. This is because the definite article and demonstrative pronoun have identical surface forms. Importantly the grammar correctly rules out 8 and 10.

5 Conclusion

In this article it has been pointed out that an important aspect of noun phrase analysis is the specification of selection restrictions for noun phrase specifiers in order to account for the combination of multiple specifiers. Danish data was presented which showed that definiteness plays an important role in this respect. This was reflected by the noun phrases containing multiple definite specifiers, and it was further shown that definiteness also plays a role among numerals and adjectives. A number of previous HPSG noun phrase accounts were discussed, and it was noted that they point towards two important properties of noun phrase structure. Firstly, specifiers must mark their projections. Secondly, a detailed classification of specifiers is required. An analysis was then presented which incorporated these properties together with a cross categorial definiteness feature to account for Danish noun phrase structure. The analysis has been implemented in the LKB system, and the results of parsing the Danish data were included, showing that the implementation, hence, the analysis, indeed provides an adequate analysis of the presented Danish noun phrases.

Footnotes

[1]Hansen (1994) provides a description of the semantics of these constructions.

[2]In Pollard & Sag (1994) specifiers are categorized as functionals, i.e. words the semantic content of which "is purely logical in nature (perhaps even vacuous)" (Pollard & Sag 1994:45). In Pollard & Sag (1994:344-393), however, it is pointed out that many specifiers have semantic content and may take their own complements and specifiers, giving rise to complex specifier phrases which do in fact contain several specifiers. It is not quite clear whether a re-classification of such specifiers as non-functionals is intended in which case the SPEC attribute would be appropriate for

both functional and substantive categories. However, what is important is that no examples are provided of complex specifiers phrases containing specifiers like articles or demonstratives, which means that multiple specifier sequences containing these are not accounted for. Even if these categories were contained in complex specifier phrases, the unaddressed problem consisting in defining head-dependent relations remains. Allegranza (1998) addresses this issue, and concludes that the establishment of such relations would be "quite arbitrary".

References

Abney, S. 1987. The English Noun Phrase in its Sentential Aspect. MA. Thesis, MIT.

Allegranza, V. 1998. Determiners as Functors: NP Structure in Italian. In S. Balari & L. Dini (eds) *Romance in HPSG*, CSLI Lecture Notes, Vol. 75. CSLI Publications: Stanford, 55-108.

Börjars, K. 1994. Feature Distribution in Swedish Noun Phrases. Phd. dissertation, Department of Linguistics, University of Manchester.

Chomsky, N. 1970. Remarks on Nominalization. In R. A. Jacobs, and P. S. Rosenbaum (eds) *Readings in English Transformational Grammar.* Waltham, MA: Ginn, 184-221.

Copestake, A. 1999. *The (new) LKB system.* Stanford University.

Delsing, L. 1993. *The Internal Structure of Noun Phrases in the Scandinavian Languages.* Malmö: Team Offset.

Hansen, E. 1994. Kvalificeret bestemthed. Unpublished manuscript, Københavns Universitet.

Jackendoff, R. 1977. $\overline{X}$ *Syntax: A Study of Phrase Structure.* Cambridge, Massachusetts: MIT Press.

Kathol, A. 1998. Determiners as Nominal Heads. In Gosse Bouma, Geert-Jan Kruijff & Richard Oehrle(eds) *Proceedings of FHCG'98*, Saarbrücken 1998, 136-143.

Kolliakou, D. 1995. Definites and Possessives in Modern Greek: an HPSG Syntax for Noun Phrases. Ph.D. thesis, University of Edinburgh.

Netter, K. Towards a Theory of Functional Heads: German Nominal Phrases. In J. Nerbonne, K. Netter and C. Pollard (eds) *German in Head-Driven Phrase Structure Grammar.* Stanford: CSLI.

Pollard, C. & Sag, I. 1994. *Head-Driven Phrase Structure Grammar.* Chicago: The University of Chicago Press.

Tonem 1 eller 2 eller 1,5?

Arild Noven (arild.noven@gri.no), Per Arne Larsen (pelarsen@gri.no),
Bente Moxness (bente.moxness@gri.no), Kolbjørn Slethei
(kolbjorn.slethei@gri.no)
Voss International Language Technology A/S

Samandrag

Artikkelen presenterer eit eksperiment utført ved Nordisk Språkteknologi A/S der ein vurderte nytten av eit nøytralt tonem som erstatning for tonem 1 eller 2 i ein norsk talesyntetisator. Arbeidshypotesen var at val av dialekt er heilt avgjerande for utfallet, og Ålesunds-dialekten vart vald avdi han har ein minimal skilnad mellom tonem 1 og 2. Eit nøytralt tonem 1,5 vart konstruert som ein interpolasjon mellom tonem 1 og 2, og 2 testsetningar, kvar med ulike kombinasjonar av tonem 1, 1,5 og 2, vart presentert for ei gruppe utvalde forsøkspersonar. Forsøkspersonane vart så bedne om å vurdera kvaliteten på testsetningane langs ein 5 punktsskala. Resultata viste at tonem 1,5 vart akseptert som fullgod erstatning for dei ekte tonema.

1. Innleiing

Det å konstruera ein talesyntetisator byr på mange ulike typar utfordringar frå dei reint tekniske til dei meir lingvistiske. Ein stor del av det prosesseringsarbeidet som vert utført frå skriven tekst til ferdig lyd, er av lingvistisk og fonetisk art. Dei skandinaviske språka har mange lingvistiske og fonetiske særtrekk som gjer eit slikt prosjekt vanskelegare enn for mange andre europeiske språk. Tonelagsskilnad, utstrekt bruk av komposita og den komplekse morfologien i slike ord er mellom dei tinga som gjer dette arbeidet krevjande.
I 1984 kom den første versjonen av ein norsk talesyntetisator – den såkalla INFOVOX-syntetisatoren - som resultat av eit samarbeidsprosjekt mellom KTH i Stockholm og Universitetet i Bergen. Under utviklinga av denne syntetisatoren, vart det klart at særleg to problemområde var avgjerande for kvaliteten, nemleg trykkplassering og val av tonem. Sidan dei aller fleste norske dialektar er tonemiske, og bruk av korrekt tonem er ein viktig faktor i ein talesyntetisator av høg kvalitet, var det avgjerande å finna reglar som styrde tildeling av rett tonem til dei aktuelle orda.[1] Imidlertid viste dette seg å vera eit ikkje-trivielt problem sidan både leksikalske og bøyingsmorfologiske faktorar spelar inn. (Sjå også Carlson *et al.* (1982) og Carlson & Granström (1986).) Tabell 1 viser litt av denne kompleksiteten [2] (Tonem 1 og 2 er markert med ' og '')

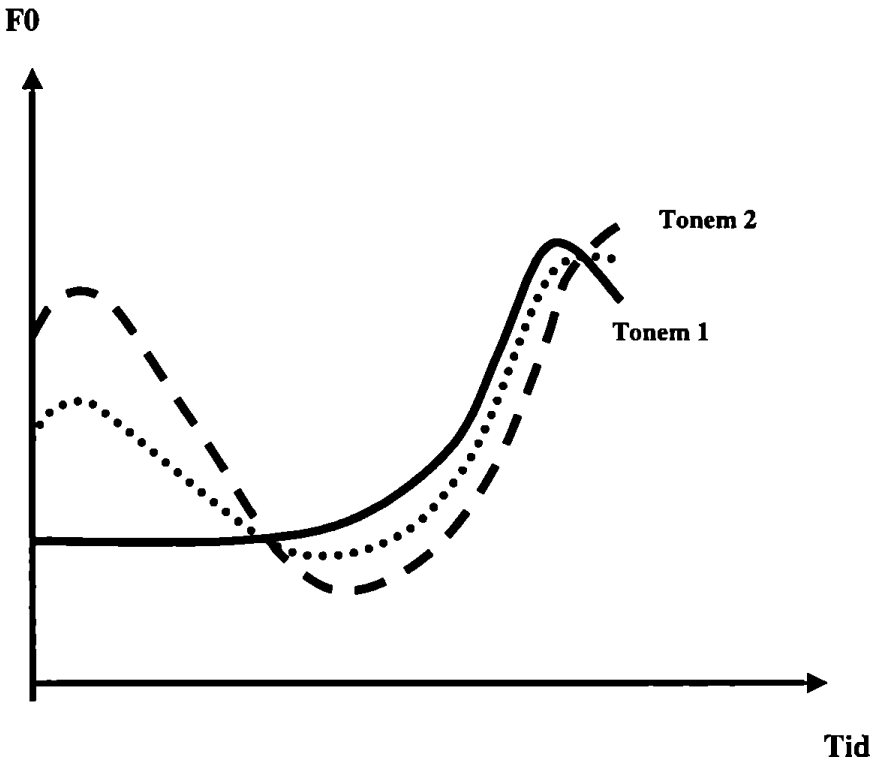
Bøyingsform	Struktur	Transkripsjon	Struktur	Transkripsjon
Infinitiv	løp-e	/"lø:pe/	hopp-e	/"hɔpe/
Presens	løp-er	/'lø:per/	hopp-er	/"hɔper/
Preteritum	løp-	/'lø:p/	hopp-et	/"hɔpet/
Ubest. sing.	løp-	/'lø:p/	hopp-	/'hɔp/
Best. sing.	løp-et	/'lø:pe/	hopp-et	/'hɔpe/
Ubest. plur.	løp-	/'lø:p/	hopp-	/'hɔp/
Best. plur.	løp-ene	/'lø:pene/	hopp-ene	/'hɔpene/
Infinitiv	kjøp-e	/"çø:pe/	hend-e	/"hene/
Presens	kjøp-er	/"çø:per/	hend-er	/"hener/
Preteritum	kjøp-te	/"çø:pte/	hend-te	/"hente/
Ubest. sing.	kjøp-	/'çø:p/		
Best. sing.	kjøp-et	/'çø:pe/		
Ubest. plur.	kjøp-	/'çø:p/	hend-er	/'hener/
Best. plur.	kjøp-ene	/'çø:pene/	hend-ene	/'henene/
Ubest. sing.	bøk-	/'bø:k/		
Best. sing.	bøk-en	/'bø:ken/		
Ubest. plur.	bøk-er	/"bø:ker/	bøk-er	/'bø:ker/
Best. plur.	bøk-ene	/"bø:kene/	bøk-ene	/'bø:kene/

Tabell 1. Tonelag 1 og 2 i høve til leksikalske og bøyingsmorfologiske faktorar.

Som det vil gå fram av denne tabellen, er det uråd å laga systematiske reglar for tildeling av tonem når ein ser på endingane –er, -en, -et og –ene. All erfaring med bruk av reglar for tildeling av tonem syner også at det er vanskeleg å finna eit "vanntett" regelsett. Dersom eit ord vert uttalt med feil tonem høyrest det unaturleg ut og kvaliteten på syntesen vert dårleg.

2. Tonem 1,5

Idéen om eit nøytralt tonem som kan erstatta tonem 1 og 2 i ein talesyntetisator var grunnlaget for ei undersøking utført av Björn Granström og Kjell Gustafson, presentert på konferansen "Nordic Prosody" i Odense i 1987, jfr. Granström & Gustafson (1987). Med utgangspunkt i austnorsk dialekt, vart det nøytrale tonemet, tonem 1,5, konstruert som ein mellomting av tonem 1 og 2 ved å interpolera F0-kurvene for desse som illustrert i fig. 1.



Figur 1. Tonem1,5 interpolert mellom tonem 1 og 2 for austnorsk.

Arbeidshypotesen i undersøkinga var at dersom eit automatisk system for tildeling av tonem fungerte dårleg, ville det vera betre om ein konsekvent brukte tonem 1,5 fordi dette tonemet ville bli oppfatta som enten tonem 1 eller 2 avhengig av kva kontekst ordet stod i. For å testa ut denne hypotesen vart det konstruert 2 basissetningar:

Basissetningar

I Du må gjenta når det skal lande
II Da tar jenta bilen ut på landet

Ut frå desse basissstningane vart det så konstruert 4 referansesetningar og 3 testsetningar:

Referansesetningar

1. Begge tonem korrekte
2. Første tonem feil

3. Andre tonem feil
4. Begge tonem feil

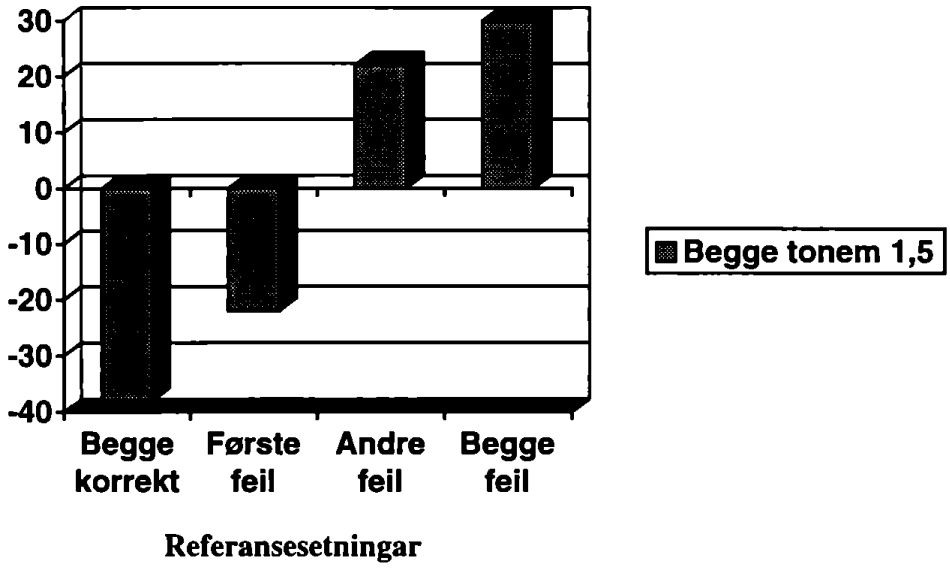
Testsetningar

1. Begge tonem 1,5
2. Første tonem korrekt, andre tonem 1,5
3. Første tonem 1,5, andre tonem korrekt

Referanse- og testsetningane vart så presenterte parvis i randomisert rekkjefølgje til ei gruppe på 14 forsøkspersonar. Dei siste 8 setningspara vart presentert ein ekstra gang i starten av testen for at forsøkspersonane skulle venna seg til lyttesituasjonen. Resultata frå desse setningane vart seinare eliminert frå testen. I eit setningspar vart første setning kalla A, og siste B. Kvar forsøksperson skulle så avgjera kva setning som høyrdest best ut langs ein 5-punkts skala:

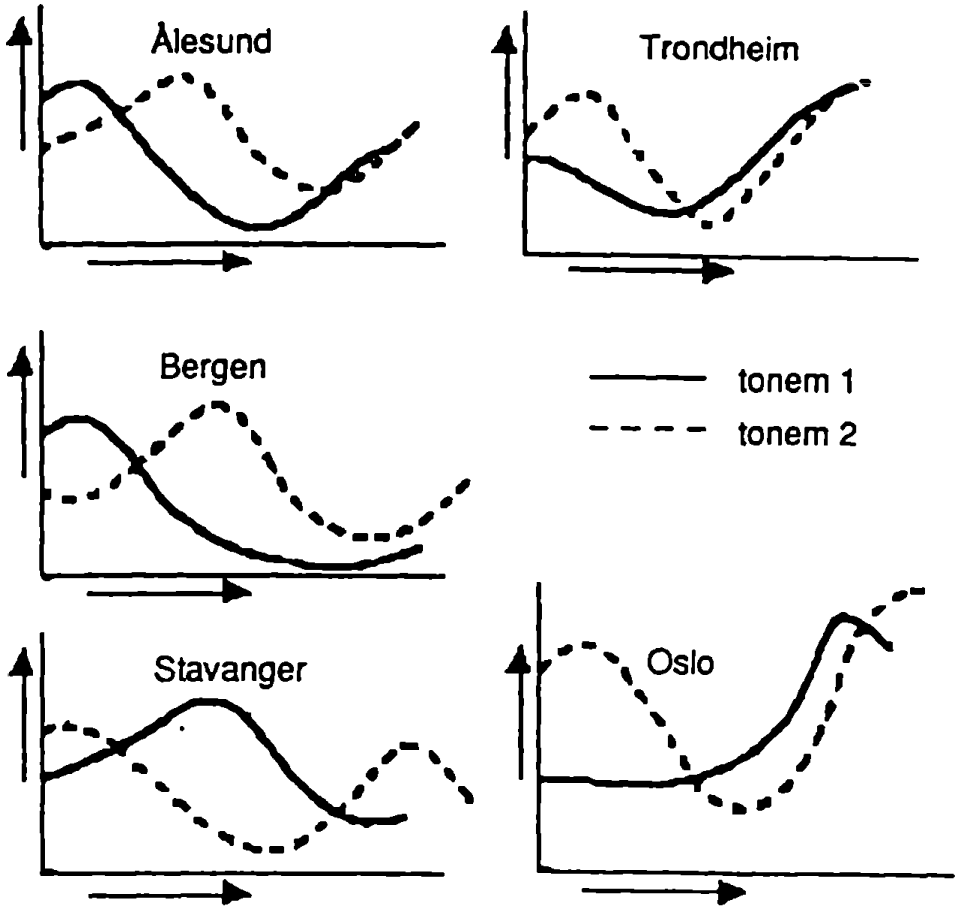
1. A mykje betre enn B
2. A litt betre enn B
3. Begge omtrent like gode
4. B litt betre enn A
5. B mykje betre enn A

I konklusjonen for denne undersøkinga heiter det at tonem 1,5 vart vurdert som eit godt kompromiss mellom korrekt og feil bruk av ekte tonem. Dette resultatet er tufta på ein implisitt føresetnad om at resultata frå dei 3 testsetningane kan og skal vektast likt. Imidlertid lyt resultata frå denne testen tolkast på litt ulike måtar avhengig av kva applikasjon dette skal brukast i. Ein mogleg applikasjon er t.d. at ein har ei unntaksliste av høgfrekvente ord som inneheld informasjon om korrekt tonelag, og at ein berre brukar tonem 1,5 dersom ordet ikkje står i denne lista. Imidlertid vert det då vanskeleg å tolka dei resultata som gjeld testsetning 2 og 3 fordi antal tilfelle der tonem 1,5 vert brukt alltid vil vera langt større enn antal tilfelle der ein kan bruka korrekt tonelag henta frå høgfrekvenslista. Ei liste av høgfrekvente ord kan berre innehelda nokre få tusen ord, og dermed gjenspeglar ikkje resultata som gjeld testsetning 2 og 3 ein realistisk applikasjon. Dersom ein skal få fram ei meiningsfull tolking av resultata frå Granström og Gustafson si undersøking, er det berre testsetningar med konsekvent bruk av tonem 1,5 som er aktuelt å vurdera resultata ut frå. Figur 2 gjev ei framstilling av resultata dersom ein samanliknar gjennomført bruk av tonem 1,5 med bruk av dei ekte tonema.



Figur 2. Vurderingar av testsetningar samanlikna med referansesetningar.
Talemålsgrunnlag: Austnorsk.

Negativ verdi på skalaen tyder altså at referansesetninga vert vurdert som betre enn testsetninga. Om ein tenkjer seg ein realistisk applikasjon der ein gitt grammatikk styrer tildeling av tonem med ein treffprosent på 50, så vil kvar av dei 4 kategoriane for referansesetningane vera like sannsynlige (p=0,25), d.v.s. at dei lyt vektast likt når ein skal gjera opp status. Om ein no brukar tonem 1,5 konsekvent, ser ein at resultatet vert dårlegare (negativ totalsum) enn om tonemtildelinga vert styrt av ein grammatikk som tek feil i 50% av tilfella.
Dette er eit ganske anna resultat enn det forfattarane ønskte å presentera i denne artikkelen. Det er grunn til å spørja kva grunnen til eit slikt negativt resultat kan vera. Om ein studerer tonemkonturane for ulike norske dialektar, er det rimeleg å tenkja at resultatet er avhengig av kva dialekt ein legg til grunn for undersøkinga. Figur 3 viser tonemkonturane for 5 norske dialektar.



Figur 3. Tonemkonturar for 5 norsk dialektar. Vertikal akse: F0. Horisontal akse: Tid. (Figuren er tilpassa frå Fintoft, K. 1970. *Acoustic Analysis and Perception of Tonemes in some Norwegian Dialects.* Oslo: Universitetsforlaget.)

Som det går fram av figur 3, har Oslo-dialekten nokså stor skilnad mellom F0-konturane for tonem 1 og 2. Særleg gjeld dette den første og trykksterke fasen av tonemrealiseringa som tilsvarar omtrent 1. halvpart av konturen. Det er denne fasen som er perseptorisk viktig når det gjeld å kunna skilja dei 2 tonema frå kvarandre, og av den grunn er Oslo-dialekten lite eigna som utgangspunkt for eit nøytralt, mellomliggjande tonem. Det er mange måtar å måla skilnad mellom tonemkonturar på, men 2 kriterium synest vera naturleg å festa seg ved:

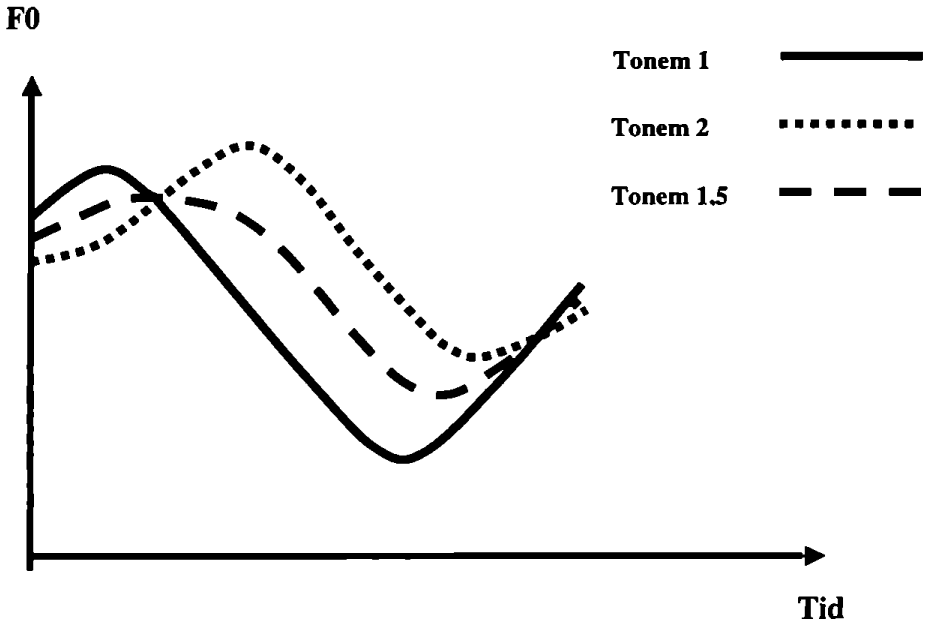
- Absoluttverdien av F0-skilnaden langs ulike tidspunkt i den 1. fasen av realiseringa
- Spektral vs. temporal konturskilnad

Målt etter begge kriteria, ser ein at Ålesunds-dialekten har ein relativt liten skilnad mellom tonemkonturane samanlikna med dei andre dialektane. Særleg tydleg vert dette dersom ein fokuserer på den temporale skilnaden: Ved å parallellforskyva konturen for

tonem 1 langs tidsaksen, vil han falla saman med konturen for tonem 2. Dette er ikkje råd å få til med tilsvarande konturar frå nokon av dei andre dialektane.

3. Nytt forsøk

Vi ønskte å undersøkja om andre dialektar enn Oslo-dialekten kunne vera betre eigna for ei tilsvarande undersøking. Ålesunds-dialekten skilde seg naturleg ut i denne samanhengen av dei grunnar som er nemnde ovanfor. Figur 4 viser tonemkonturane for Ålesunds-dialekten der også konturen for tonem 1,5 er teikna inn.



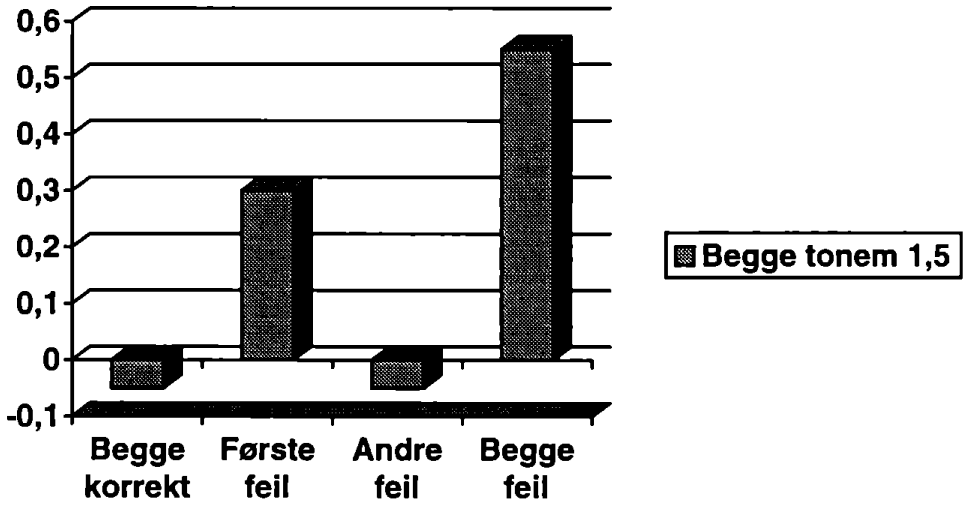
Figur 4. Tonemkonturar for Ålesunds-dialekten.

Det går tydleg fram av figur 4 at avstanden frå konturen til tonem 1,5 og til ein av dei andre konturane er klart mindre enn tilsvarande avstand i fig. 1. Undersøkinga vart utført etter same lest som hos Granström og Gustafson, men med nokre viktige modifikasjonar:

- Fastsetjing av testintern 0-linje v.h.a. kontrollpar som var akustisk identiske
- Bruk av standard statistiske metodar (Wilcoxons signed rank test)
- Statistisk kontroll av likeverdighet m.o.t. effekten av tonem 1,5 for dei 2 basissetningane I og II
- Presentasjonsrekkefølge går ut som variabel

Ei gruppe på 13 forsøkspersonar vart sett saman av personar med fonetisk og lingvistisk bakgrunn, d.v.s. med både teoretisk og praktisk skulering i begge disiplinar. Alle gjennomgjekk først ein test der det vart undersøkt om dei hadde tilstrekkeleg "kompetanse" i å diskriminera tonelag på bakgrunn av ein lyttetest. Resultata frå 3 av

desse personane måtte utelatast frå testmaterialet fordi dei ikkje klarte den innleiande testen.
Fastsetjing av testintern 0-linje er viktig for å få til ei innbyrdes kalibrering av dei ulike forsøkspersonane. Det var nemleg store individuelle skilnader mellom forsøkspersonane m.o.t. korleis dei oppfattar setningar med akustisk identiske stimuli. Ved å spela av 6 setningspar som kvar inneheldt 2 akustisk identiske setningar, kunne ein registrera slike individuelt betinga avvik frå "0" som testresultat. Slike avvik må då skuldast eigenskapar hos den einskilde person og ikkje eigenskapar ved testmaterialet. Ein forsøksperson hadde 0,83 i snitt medan andre hadde –0,16 i snitt på denne testen. Det er difor viktig å kalkulera inn denne skilnaden i den vidare utrekninga elles risikerer ein at faktorar som er knytte til den einskilde forsøkspersonen øydelegg resultatet av undersøkinga.
Ei psykoakustisk undersøking som denne lyt gjennomførast med standard statistiske metodar for å vera vitskapleg gyldig. Den største veikskapen ved undersøkinga til Granström og Gustafson er at dei dreg konklusjonar av talmaterialet utan å underkasta det vanlege statistiske metodar for hypoteseprøving. Sett frå ein vitskapleg synsvinkel vert dermed konklusjonane ugyldige, og det er ikkje råd å seia om dei resultata dei kjem fram til skuldast faktiske skilnader eller tilfeldige samantreff. Av den grunn har vi heller ikkje gjort ei direkte samanlikning mellom resultata frå dei to undersøkingane.
Det er ikkje sjølvsagt at dei 2 basissetningane "Du må gjenta når det skal lande" og "Da tok jenta bilen ut på landet" er likeverdige m.o.t. effekten av tonem 1,5. Dette lyt også undersøkjast med statistiske metodar. Resultatet viste ingen signifikant skilnad mellom desse setningane på dette området[3], og det aritmetiske gjennomsnittsresultatet for desse 2 basissetningane kan difor nyttast for kvar av dei 24 moglege setningspara i testen.
Figur 5 viser resultatet av undersøkinga. Berre dei resultata som tilsvarar resultata i figur 2 er tekne med.



Figur 5. Vurderingar av testsetningar samanlikna med referansesetningar.
Talemålsgrunnlag: Ålesund bymål.

Som det går fram av figur 5, kjem tonem 1,5 best ut i 2 tilfelle, medan dei ekte tonema kjem best ut i dei 2 andre tilfella. Imidlertid er ingen av desse resultata statistisk signifikante på 5% nivå, sjølv om tendensen er klart i favør av tonem 1,5. På 10% forkastningsnivå er det siste resultatet (tonem 1,5 kontra begge tonem feil) så å seia signifikant.

4. Konklusjon

Bruk av tonem 1,5 er like bra eller betre som bruk av ekte tonem i alle setningspara der konsekvent bruk av tonem 1,5 vert samanlikna med bruk av ekte tonem.
Val av dialekt er avgjerande for utfallet av ei slik undersøking.

Fotnotar

[1] Problemet med korrekt tildeling av tonem er like aktuelt for svensk talesyntese, sidan dei aller fleste svenske dialektar også er tonemiske. Imidlertid er problema litt færre i svensk sidan nestan alle kompositaord får tonem 2, mens det same ikkje er tilfelle for norsk.

[2] Tabellen er utarbeidd av Kjell Gustafson og presentert i artikkelen "Some innovations in a Norwegian text-to-speech system", pp. 81-85 trykt i *Quarterly Progress and Status Report*, Stockholm, 1/1989.

[3] Wilcoxon signed-rank, tosidig testing. Minste verdi: p=0,32.

Litteratur

Carlson, R., Granström, B. & Hunnicutt, S. 1982. A multi-language text-to-speech module. *Proceedings ICASSP- Paris*, 3,1604-1607.

Carlson, R. & Granström, B. 1986. Linguistic processing in the KTH multi-lingual text-to-speech system, *Proceedings ICASSP- Tokyo*. 4, 2403-2406 .

Granström, B. & Gustafson, K. 1987. Toneme 1 ½ in a Norwegian Text-to-Speech System. *Nordic Prosody IV, Papers from a symposium*, Odense: Odense University Press.

Syntactic Analysis and Error Correction for Danish in the SCARRIE project

Patrizia Paggio
Center for Sprogteknologi
patrizia@cst.ku.dk

May 10, 2000

Abstract

This paper reports on work carried out at CST in Copenhagen to develop the Danish version of the SCARRIE prototype, addressing in particular the issue of how a form of shallow parsing is combined with error detection and correction to treat context-dependent spelling errors. The paper describes the corpora used to develop the system, and shows some preliminary evaluation results.

1 The SCARRIE project

SCARRIE was a EU-funded collaborative project, the purpose of which was to develop a high-quality proof-reading tool for the Scandinavian publishing industry. The consortium consisted of partners from Sweden, Norway, and Denmark[1]. The project terminated in the spring of 1999, and resulted in the development of three prototypes covering Norwegian, Swedish and Danish. Although the three prototypes all address the same basic issue, the functionality they provide differs slightly to reflect language specific needs as well as different research interests and expertise in the groups. In this paper, we shall deal with the Danish SCARRIE, and focus on its grammar checking component. For a description of the Norwegian prototype, see (de Smedt and Rosen, this volume).

SCARRIE builds on CORRie (Vosse 1992) (Vosse 1994), a proof-reading system originally developed for Dutch and distributed by Stichting Cognitieve Technologie (the Netherlands), who acted as a subcontractor in the project. The project has adapted the system to comply with the language specific features of the languages covered, and with the requirements of the project's end users. The system processes text

in batch mode and produces an annotated output text where errors are flagged and replacements suggested where possible. Text correction is performed in two steps: first the system deals with spelling errors and typos resulting in invalid words, and then with grammar errors.

2 The Danish prototype

Localisation of the system to the Danish language has mainly consisted in the development of a set of lexical and grammar resources. These include first of all a dictionary of 251,000 domain-relevant word forms that have been derived from a collection of 68,000 newspaper articles. From the same text collection we have also extracted a separate list of 717 idioms. The list is used to identify multi-word expressions such as complex prepositions, or idioms including words that would be invalid in isolation (e.g. *carte blanche*). Both dictionary and idiom list were developed through a cooperation between CST and the Danish language and literature society (det Danske Sprog- og Litteraturselskab).

Another important component is the compound analysis grammar, a set of regular expressions covering the most common types of compound nominals in Danish. This is an important feature, as in Danish compounding is very productive, and compounds are written as single words.

Words which the system cannot find in the dictionary or the idiom list, or analyse as compound forms, or assign the label of proper name, are taken to be spelling errors. The system flags them as such and tries to suggest a replacement. The algorithm used is based on *trigram and triphone analysis* (van Berkel & de Smedt 1988), and takes into account the orthographic strings corresponding to the invalid word under consideration and its possible replacement, as well as the phonetic representations of the same two words. Phonetic representations are generated by a set of grapheme-to-phoneme rules (Hansen 1999) the aim of which is to assign phonetically motivated misspellings and their correct counterparts, identical or similar phonetic representations.

The last lingware component developed for the Danish prototype is the phrase structure grammar, which is used by the parser to identify context-dependent spelling errors (from hereafter grammar errors). Parsing results are passed on to a corrector to find replacements for the errors found. The parser is an implementation of the Tomita algorithm with a component for error recognition whose job is to keep track of error weights and feature mismatches as described in (Vosse 1991). Each input sentence is assigned the analysis with the lowest error weight. As we shall see in more detail below, the system can treat grammar errors of

two different kinds, i.e. feature mismatches and structural errors. In the case of a feature mismatch, the system tries to find the correct form of the misspelt word by overriding the offending feature, and by looking for an alternative word form in the dictionary. In the case of a structural error, on the other hand, specific error rules are applied to parse the incorrect input and an error message is generated.

3 The errors

To define the coverage of the system, the project has assembled corpora of parallel raw and proofread texts for the three languages involved. The Danish corpus consists of newspaper and magazine articles published in 1997 for a total of 270,805 running words. The articles have been collected in their raw version, as well as in the edited version provided by the publisher's own proofreaders. Although not very large in number of words, the corpus consists of excerpts from 450 different articles to ensure a good spread of lexical domains and error types. The corpus has been used to define the coverage of the grammar and to extract test data.

The errors occurring in the corpus have been analysed according to the taxonomy in (Rambell 1997). Figure 1 shows the distribution of the various error types into the five top-level categories of the taxonomy. As can be seen, grammar errors account for 30% of the errors. Of these, 70% fall into one of the following categories (Povlsen 1998):

- Too many finite verbal forms or missing finite verb
- Errors in nominal phrases:
  - agreement errors,
  - wrong determination,
  - genitive errors,
  - errors concerning pronouns;
- Split-ups and run-ons.

Another way of grouping the errors is by the kind of parsing failure they generate. As mentioned earlier, we can make a distinction between feature mismatches and structural errors. Agreement errors are typical examples of feature mismatches. In the following nominal phrase, for example:

(1) de *interessant projekter
(the interesting projects)

Error type	No.	%
Context independent errors	386	38
Context dependent errors	308	30
Punctuation problems	212	21
Style problems	89	9
Graphical problems	24	2
Total	1019	100

Figure 1: Error distribution in the Danish corpus

the error can be formalised as a mismatch between the definiteness of the determiner *de* (the) and the indefiniteness of the adjective *interessant* (interesting). Adjectives have in fact both an indefinite and a definite form in Danish.

The sentence below, on the other hand, is an example of structural error.

(2) i sin tid *skabet han skulpturer over atomkraften
(during his time wardrobe/created he sculptures about nuclear power)

Since the finite verb *skabte* (created) has been misspelt as *skabet* (the wardrobe), the syntactic structure corresponding to the sentence is missing a verbal head.

Run-ons and split-ups are structural errors of a particular kind, having to do with leaves in the syntactic tree. In some cases they can only be detected on the basis of the context, because the misspelt word has the wrong category or carries some other grammatical feature that is incorrect in the context. Although the system has a facility for identifying and correcting split-ups and run-ons based on a complex interaction between the dictionary, the idiom list, the compound grammar and the syntactic grammar, this facility has not been fully developed yet, and will therefore not be described any further here. More details can be found in (Paggio 1999).

The next section describes the way in which agreement errors in NPs and structural errors in verb groups are dealt with in the grammar, and explaines how the treatment of these errors fits in with the general analysis strategy adopted in the grammar.

4 The grammar

The grammar is expressed in an augmented context-free grammar formalism consisting of rewrite rules where symbols are associated with

features. It is also possible to add error weights to both rules and individual features, and to specify error messages. The rules are applied by unification, but in cases where one or more features of a given word do not unify with relevant features in a grammar rule, the offending features can be overridden.

Two kinds of rules may be used, "normal" rules describing the valid structures of the language, and "error" rules describing invalid structures. Thanks to the feature overriding mechanism, however, normal rules can also analyse sentences containing feature mismatch errors.

4.1 Feature mismatches

For example, consider the following rule, which is intended to account for definite nominal phrases:

```
NP(def Gender PersNumber) ->
        Det(def Gender PersNumber)
        AP(def _ _)
        N(indef Gender:2 PersNumber)
```

The rule is used to analyse NPs consisting of a definite determiner, an adjective phrase and a noun. Determiner and adjective bear the feature "def" for *definite*, whereas the noun bears the feature "indef" for *indefinite*. This reflects the fact that determiners, adjectives and nouns all inflect for definiteness in Danish, but the noun has to be in the indefinite form if preceded by a determiner. Furthermore, the three nodes must share values for *gender* and *person/number*, as indicated by the capitalised variables "Gender" and "PersNumber".

The rule will parse a correct definite NP such as:

(3) de interessante projekter
(the interesting projects)

but also

(4) de *interessant projekter
(5) de interessante *projekterne

Both (4) and (5) violate the definiteness constraints, since the adjective is indefinite in (4), and the noun is definite in (5). The feature overriding mechanism makes it possible for the system to suggest *interessante* as the correct replacement in the former case, and *projekter* in the latter. What happens is that the parser selects the rule as applicable because the syntactic backbone matches the input, but detects a violation in one of the features. It then overrides the violating feature on the incorrect word and looks for a suitable replacement by searching

for alternative forms of the same lemma in the dictionary. The resulting analysis carries an error weight generated by the overriding operation.

Weights are used to control rule interaction as well as to establish priorities among features that may have to be overridden. For example in our NP rule, a weight has been attached to the Gender feature in the N node. The weight expresses the fact that it costs more to override gender on the head noun than on the determiner or adjective. The reason is that if there is a gender mismatch, the parser should not try to find an alternative form of the noun (which does not exist), but rather override the gender feature either on the adjective or the determiner.

4.2 Structural errors

Error rules are very similar to normal rules, the only difference being that they have to be associated with an error weight and an error message.

The purpose of the weight is to ensure that error rules are applied only if normal rules are not applicable. Error messages serve two different purposes. If they are preceded by a question mark, they only appear in the log file for the developer to trace the analysis process. Otherwise, they are shown to the end user when the rule they are associated with has been applied. In other words if an error rule is applied to analyse a sentence, the system will not look for a replacement, but present the user with an error message indicating the kind of grammatical failure that has been observed.

The following is an error rule example.

```
VGroup(_ finite Tense) ->
       V(_ finite:4 Tense)
       V(_ finite:4 _)
       "Sequence of two finite verbs":4
```

A weight of 4 is attached to the rule as a whole, but there are also weights attached to the "finiteness" feature on the daughters: their function is to make it costly for the system to apply the rule to non-finite forms. In other words, the feature specification "finite" is made difficult to override to ensure that it is indeed a sequence of *finite* verbal forms the rule applies to and flags.

The rule will for example parse the verbal sequence in the following sentence:

(6) Jeg vil *bevarer (bevare) min frihed.
(*I want keep my freedom)

As a result of parsing, the system in this case will not attempt to correct the wrong verbal form, but issue the error message "Sequence of two finite verbs".

In many cases, in fact, it may be quite difficult to suggest a correction for a wrong verb sequence, since there may be several reasonable ways of amending it, some of which imply more than just replacing one form with another.

To sum up, error rules can be used to describe an error explicitly and to issue error messages. However, so far we have made very limited use of them, as controlling their interaction with normal rules and with the feature overriding mechanism is not entirely easy. To this sparse use of error rules corresponds, on the other hand, an extensive exploitation of the feature overriding mechanism. This strategy allows us to keep the number of rules in the grammar relatively low, but relies on a careful manual adjustment of the weights attached to the various features in the rules.

4.3 Shallow analysis

In the current version of the grammar, only the structures relevant to the error types we wanted the system to deal with – nominal phrases and verbal groups – are accounted for in detail. The analysis produced is thus a kind of shallow syntactic analysis where the various sentence constituents are attached under the topmost S node as fragments. This choice was made for two reasons. Firstly, we wanted the system to target the error types represented in our corpus to tailor its functionality to the needs of our end users. Secondly, we did not want to impair the system's efficiency by striving for too complex a model of syntactic analysis.

Below, we show the rules (the features attached to the various categories have been removed here for the sake of exposition) implementing the fragment strategy just mentioned:

```
S -> Fragments VGroup Fragments
S -> Fragments VGroup
S -> InvVGroup Fragments
Fragments -> Fragment
Fragments -> Fragment Fragments
Fragment -> NP
Fragment -> PP
Fragment -> AdvP
Fragment -> ...
```

As can be seen, a sentence is built up of a verb group possibly preceded and followed by one or more fragments, in turn analysed as either NPs, PPs and so on. The internal structure of verb groups - including possible wrong structures - is specified in a number of rules rewrit-

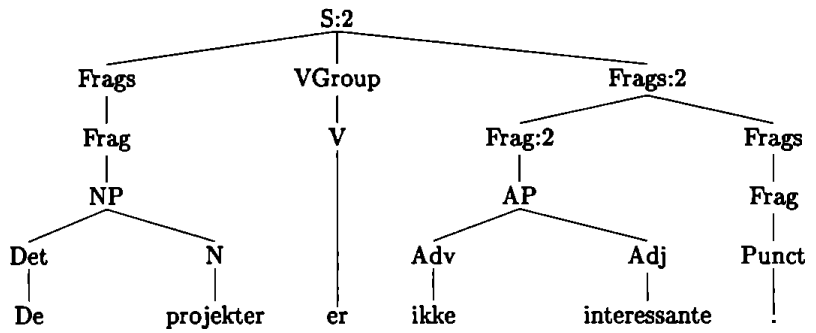


Figure 2: A parse tree

ing VGroup. InvVGroup in the third sentence rule above, accounts for subject-main verb inversion in interrogative sentences.

There are cases, of course, in which attaching a constituent directly under the S node does not enable the system to spot an error for which we would expect a flag. Adjective phrases are an example. Since agreement errors in nominal phrases as we saw are rather frequent in the SCARRIE database, we wanted the system to be able to identify and correct them. Therefore, APs can also be analysed as nominal modifiers by the NP rules. To indicate then that the fragment analysis is not optimal (it should only be resorted to when the adjective is not part of a nominal phrase), it is associated with an error weight, as well as a system-internal error message (invisible to the end user):

```
Fragment -> AP "?Fragment AP rule":2
```

The weight penalises parse trees built by applying the rule. However, in lack of a better solution, the rule is triggered e.g. to analyse the AP in the following sentence:

(7) De projekter er ikke interessante.
(Those projects are not interesting)

The parse tree produced by the system is shown in Figure 2. Note that the error weight introduced by the Fragment AP rule is percolated up to the top S node.

5 Evaluation and Conclusion

The evaluation methodology adopted in the project capitalises on the fact that we had access to a set of parallel unedited and proofread texts

(see (Paggio & Music 1998)). This made it possible to develop a tool that compares the results obtained by the system with the corrections suggested by the publisher's human proofreaders. The tool derives *recall* measures (lexical coverage as well as coverage of errors), a *precision* measure (percentage of correct flaggings), as well as *suggestion adequacy* measures (hits, misses and no suggestions offered). The same automatic procedure was used to evaluate the system during development, and to validate it at the user site. Testing was done on constructed test suites displaying examples of the errors targeted in the project and with text excerpts from the parallel corpora.

The results obtained on the test suites are very positive, especially with regard to the treatment of grammar errors. More extensive testing (see (Paggio to appear) for more details) has shown that when the system is run on a text, error coverage decreases especially because of punctuation and other stylistic matters not treated in the project. There are also, however, agreement errors which go unnoticed, mainly due to the imprecision introduced by the fragment analysis approach. A large number of the false flags produced is due to the grammar's limited coverage. In particular, genitive phrases, which are not treated at the moment, are often the cause of wrong NP analyses.

Considering the fact that relatively little time was spent on grammar development in the project's lifetime, we consider the results obtained encouraging. There is, however, space for improvement, especially with regard to extending the coverage of the grammar.

Notes

[1]Main contractors were: WordFinder Software AB (Sweden), Center for Sprogteknologi (Denmark), Department of Linguistics at Uppsala University (Sweden), Institutt for lingvistikk og litteraturvitenskab at the University of Bergen (Norway), and Svenska Dagbladet (Sweden). A number of subcontractors also contributed to the project. Subcontractors in Denmark: Munksgaard International Publishers, Berlingske Tidende, Det Danske Sprog- og Litteraturselskab, and Institut for Almen og Anvendt Sprogvidenskab at the University of Copenhagen (Denmark).

References

Hansen, P. M. (1999). Grapheme-to-phoneme rules for the Danish component of the SCARRIE project. H. E. Thomsen & S. Kirchmeier-Andersen, eds, *Datalingvistisk Forenings årsmøde 1998 i Købehavn, Proceedings*, number 25 in 'LAMBDA'. Institut for datalingvistik, Handelshøjskolen i København, 79–91.

Paggio, P. (1999). Treatment of grammatical errors and evaluation in SCARRIE. H. E. Thomsen & S. Kirchmeier-Andersen, eds, *Datalingvistisk Forenings årsmøde 1998 i København, Proceedings*, number 25 in 'LAMBDA'. Institut for datalingvistik, Handelshøjskolen i København, 65–78.

Paggio, P. (to appear). Danish grammar checking in SCARRIE. *Proceedings of ANLP 2000.* Seattle, Washington.

Paggio, P. & Music, B. (1998). Evaluation in the SCARRIE project. *Proceedings of the First International Conference on Language Resources & Evaluation.* Granada, Spain, 277–282.

Povlsen, C. (1998). Three types of grammatical errors in Danish. Technical report. Copenhagen: Center for Sprogteknologi.

Rambell, O. (1997). Error typology for automatic proof-reading purposes. Technical report. Uppsala: Uppsala University.

van Berkel, B. & de Smedt, K. (1988). Triphone analysis: a combined method for the correction of orthographical and typographical errors. *Proceedings of the 2nd conference on Applied Natural Language Processing.* ACL, Austin, 77–83.

Vosse, T. (1991). Detection and correction of morpho-syntactic errors in shift-reduce parsing. R. Heemels, A. Nijholt & K. Sikkel, eds, 'Tomita's Algorithm: Extensions and Applications', number 91-68 in *Memoranda Informatica.* University of Twente, 69–78.

Vosse, T. (1992). Detecting and correcting morpho-syntactic errors in real texts. *Proceedings of the Third Conference on Applied Natural Language Processing.* Trento, Italy, 111–118.

Vosse, T. G. (1994). The Word Connection - Grammar-based Spelling Error Correction in Dutch. PhD thesis. Rijksuniversiteit at Leiden, the Netherlands. ISBN 90-75296-01-0.

Designing a System for Swedish Spoken Document Retrieval

Botond Pakucs[1,2] **Björn Gambäck**[1,3]

botte@speech.kth.se gamback@sics.se

[1] Information and Language Engineering
Swedish Institute of Computer Science
Box 1263
S-164 29 Kista, Sweden

[2] Centre for Speech Technology
Royal Institute of Technology
Drottning Kristinas väg 31
S-100 44 Stockholm, Sweden

[3] Computational Linguistics
University of Helsinki
P.O. Box 4
SF-00014 Helsinki, Finland

Abstract

It is only during the last few years that attention has started to shift from pure text-based retrieval towards other media. Information retrieval from spoken documents is analogous to text-based retrieval; however, accessing audio documents causes some extra problems, in particular with respect to document segmentation, choice of indexing features, and robustness. In addition, retrieval of documents in Swedish, like most non-English languages, adds the extra dimension of morphology; also, when analysing spoken Swedish data, prosodic patterns have to be taken into account. In this paper we introduce SIREN, the Swedish Information Retrieval Engine, a very flexible, modular IR system which has been designed with a specific eye towards these issues.

1 Introduction

The field of information retrieval (IR) has been moving steadily forward for several decades. During the 90's we have seen several major break-throughs. Until recently, most of the work has been focused on texts; not only has most of the material processed been in text format, but even when other media such as audio and video have been considered in a system, text has been the primary concern. Consequently, most multimedial retrieval systems are modifications of existing text-based IR systems, disregarding the particular problems caused by the new media types. However, the amount of material in other formats increase all the time, increasing the need for tools that handle this information both from a system-oriented and a user-oriented perspective. An important issue for information management is how to represent these objects and collections of objects to best support retrieval. Another issue is to let users search in several modalities. An example would be when a person wants to search a news database: the database contains both news in text format and audio sequences of spoken material. These are stored and indexed in different ways, but the user wants to search both archives at the same time.

The term 'Spoken Document Retrieval' (SDR) has, in itself, rendered some confusion. We will use it exclusively for the particular case of information retrieval, when the information

is to be retrieved from large volumes of spoken documents. Thus, what media the query itself is formulated in is of no importance to us here; only the format from which the sought information is to be accessed. In Section 2 we review the difficulties caused by trying to access multimedial documents, in particular audio documents, and some previous attempts to overcome them. We are building a flexible toolkit specifically designed to allow for different approaches to addressing these difficulties and for handling data in different types of media. The toolkit is functional but still open to improvement; however, Section 3 discusses the underlying design philosophy.

2 Problematic Issues

An obvious difference between information retrieval from written and spoken documents is that the written ones actually are physical documents, while the boundaries of the spoken ones, in contrast, have to be decided on inside larger audio files possibly containing quite diverse information. We will start out by looking at this segmentation issue and also at some other common problems in non-text-based retrieval, in particular with respect to choice of indexing features and robustness.

2.1 Document Segmentation

Many research projects, such as the ones in the TREC SDR Track (Garofolo *et al.*, 1997), have left document segmentation aside and instead manually tagged and segmented the documents. A straight-forward, automatic approach is to break down the audio documents into equally long time slices (windows) and treat the windows as individual documents (Schäuble and Wechsler, 1995; Smeaton *et al.*, 1998).

However, the linguistically most correct solution would be to automatically achieve meaningful segmentation of the audio documents through analysis of the prosodic information inherent in the audio documents, as done by Falk (1997). Unfortunately, not much work has been done on prosody in speech recognition. So far, the major success story was in the German spoken dialogue machine translation system *Verbmobil* (Bub *et al.*, 1997), where prosodic information was used in the semantic processing and transfer steps in order to produce correct translations (Lieske *et al.*, 1997). The input to a spoken dialogue system is structured in terms of turn-taking in the dialogue. Syntactic and semantic analyses can be assigned to meaningful linguistic entities at the clausal or phrasal level, but first the turn has to be segmented into a sequence of linguistically credible segments. The prosodic indications of clausal boundaries are crucial to this process. The prosody module of the *Verbmobil* system annotates the word lattices which have been output from the recogniser with three different kinds of prosodic information: sentence modality, phrase boundaries, and stress (Hess *et al.*, 1996).

Stolcke *et al.* (1999) introduces a combined probabilistic and prosody-based approach for recognising topic and sentence boundaries in speech. They employ a hidden Markov model (HMM) to find the most likely states given the recognised words and prosody (duration of pauses, etc., as well as pitch level, that is, $F0$ patterns; see below). In a first step the speech signal is "chopped" into small "sentence" segments which are assumed to belong to one topic each. In a second step the sentences are combined into continuous stretches

belonging to one topic so that the sentence boundaries are classified according to whether they represent a topic shift or not (Hakkani-Tür *et al.*, 1999).

A somewhat similar but purely probabilistic segmentation technique is used by Delacourt *et al.* (1999) for recognising which persons are engaged in a conversation. They try to segment the speech data at every point in which a speaker change occurs. This is done in a two-step process where the first step applies a measure function reflecting how similar two adjacent segments are (in this case with respect to melcepstral coefficients). Dissimilarity indicates speaker change. In the second step a Bayesian likelihood criterion is applied to each changing point candidate to verify the results of the first step.

The prosodic issues are also very important when analysing spoken data from a tonal-type language like Swedish; the stress patterns, in particular, simply have to be taken into account. Within an utterance, the frequency may take on four relative values depending on the accent level of a phoneme. This may be defined either in absolute terms (due to, for example, the sentence accent, or the stress level within a word), or in terms relative to the frequency of the preceding phoneme.

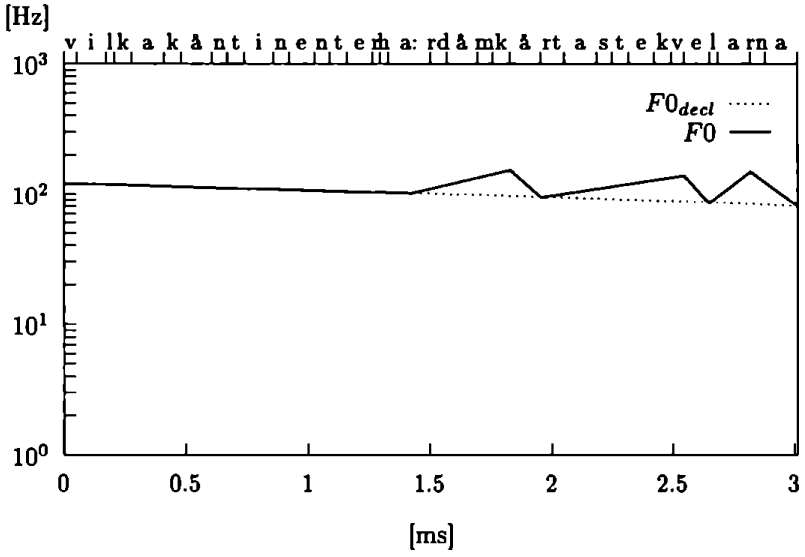


Figure 1: $F0$ and $F0$-declination functions

As described in more detail in, for example, Gambäck *et al.* (1995), the fundamental frequency contour ($F0$) is produced by superimposing the $F0$-variation on the $F0$-declination. The time points for the vocal chord excitations are then calculated from the resulting frequency function. The relation between $F0$, the variation and the declination is defined as

$$F0(t) = F0_{decl}(t) * F0_{var}(t)$$

As a concrete example, take the values that would be produced for the sentence *Vilka kontinenter har de kortaste kvällarna?*. Suppose that $F0_{decl}(t)$ starts at 120 Hz and ends at 80 Hz. $F0(t)$ is 1.8 times the declination at the time point $t = 2817$ ms and 1.6 times the declination at $t = 1827$ ms and $t = 2544$ ms. $F0(t)$ returns to the declination curve at $t = 1953$ ms, $t = 2649$ ms and $t = 3013$ ms. The function thus gets the form shown in Figure 1.

2.2 Indexing Features

In text-based information retrieval methods, the most obvious and frequently used indexing features are the words. This approach is also the dominating one in spoken document retrieval systems. However, in spite of rapidly improving speech recognition techniques, automatically produced transcriptions contain a multitude of errors and operate with limited dictionaries.

2.2.1 Word-based methods

For small domains, it is possible to select *a priori* fixed sets of keywords for recognition and indexing. This is, however, not feasible if large and diverse domains are covered. In such situations, a possible solution is to use a large-vocabulary word-based speech recogniser to convert the audio data to text and then filter the transcriptions through a language understanding system to reduce the recognition errors, as in the CMU *Informedia* project (Hauptmann, 1995). Still, when dealing with large and diverse domains of spoken documents, there are just too many words for current speech recognition systems to handle efficiently. In addition, out-of-vocabulary (OOV) words tend to be proper names or technical terms which generally have low frequencies in document collections. Thus, it is difficult to train speech recognisers for them, even though these word types are very common in user queries. This has lead to the investigation of subword unit approaches.

2.2.2 Subword unit approaches

The drawback with the use of subword indexing features is that elaborated information retrieval and linguistic techniques, such as stop word elimination and morphological analysis, are not suitable. This is specially cumbersome when the spoken information is in a language with more serious morphological properties than English. However, Ng and Zue (1997) showed that with appropriate subword units, it is possible to achieve performance comparable to that of word-based retrieval methods, if the underlying phonetic units are recognised correctly. Both syllable- and phoneme-based indexing features have been tried.

Syllabic units as indexing features was proposed by Glavitsch and Schäuble (1992). These units are composed of non-overlapping, variable-length letter sequences, CVC-features (where V and C stand for the maximum sequence of vowels and consonants, respectively, within a specific word). The syllabic units can be generated with a rule-based system (Glavitsch *et al.*, 1994). A problem with this is that the CVC-features are not based on acoustic data and do not take the characteristics of the recognition system into consideration.

n-grams, that is, overlapping, *fixed-length* phonetic sequences are the most straightforward phoneme-based subword units. With large enough length-units n, cross-word features can be captured. As indicated by, for example, Ng and Zue (1997), trigrams are normally long enough to capture information sufficient for indexing, while not being so long that they corrupt retrieval performance.

m-grams, non-overlapping, *variable-length* phonetic sequences with some maximum length m, can be discovered automatically by applying iterative unsupervised learning algorithms. Wechsler (1995) used an efficient algorithm to find variable length phone sequences for indexing. After frequency analysis, low frequency sequences were discarded, while high frequency sequences were extended. In this way an indexing vocabulary covering most parts of the phonetic transcription was obtained.

bclass, broad phonetic class sequences, are derived via unsupervised hierarchical clustering of the original phones. Acoustically similar phones are grouped into the same class, so that some of the typical speech recognition errors can be avoided. Ng and Zue (1997) claimed that even after collapsing the number of phones from 41 down to 20 bclasses, enough information is preserved to perform reasonable retrieval: Given perfect recognition, they got a precision of 0.82.

2.3 Robustness

The robustness issues facing spoken document retrieval systems are similar to when text retrieval systems encounter optical character recognition errors in scanned text documents. Thus, even given an ideal speech recognition component, low quality audio data will still effect the speech recognition accuracy and thus the retrieval performance. Accordingly, tolerance of the retrieval models to different disturbing factors such as noisy speech, conversational speech, telephone bandwidth speech, and multiple speakers is important. Quite a few solutions have been proposed to cope with speech recognition errors caused by low quality audio data; however, in the area of robust methods for spoken document retrieval, there is still much left to be done.

In word-based systems, OOV-words can cause significantly weaker retrieval results. THISL (Abberley *et al.*, 1998) applies query-time word-spotting based on posterior probability estimates derived from the recurrent network acoustic model, in order to detect index terms not in the recogniser's vocabulary. Jones *et al.* (1996) used a continuous-speech large vocabulary recognition system in combination with the phone-lattice-based word-spotting method of James (1995). They showed that the two methods are complementary and work best in combination.

In phoneme-based solutions, one possibility is to expand the query with errorful variants of the original terms, in order to improve the chance of matching wrongly recognised terms. A similar method is to expand the spoken document representation by including high scoring recognition alternatives to increase the retrieval precision. Ng (1998) tried to combine the various types of information captured by the different subword unit representations. He obtained a marginal improvement in retrieval performance by using n-best recognition hypotheses and linear combination of the individual retrieval scores obtained with different subword unit methods.

Crestani and Sanderson (1997) aimed at taking advantage of some particular features of the automatic transcriptions. They used the *Abbot* speaker independent continuous speech recognition system which associates a measure of uncertainty to each word it recognises (Robinson *et al.*, 1996). These measures were used in a probabilistic term frequency weighting scheme for improving retrieval efficiency. The results were encouraging, even though the performance was very bad on some particular queries. Later, they attempted

to merge the transcriptions produced by multiple recognisers and slight improvements in retrieval performance were achieved (Sanderson and Crestani, 1998).

2.4 Retrieval result presentation

Presenting multimedial retrieval results to the users introduces some new problems into the IR field. Determining whether a retrieved text document contains an answer to a query is quickly done by simple browsing. However, browsing information objects such as TV or radio news broadcasts can be rather cumbersome due to the sequential nature of the video/audio files.

Having decided on what piece of information to retrieve, we are still far from having decided on *how* to present the information. Most interfaces to text-based retrieval systems simply show the user a list of documents, sorted according to some kind of relevance measure, which often is incomprehensible to the user. A notable exception is the SICS–Telia prototype *Easify* where search results are presented as matrixes, rather than lists (Bretan *et al.*, 1998). *Easify* applies a machine learning method to create collection specific stylistics for genre prediction together with rapid topical clustering in order to represent documents as members of topically and stylistically homogeneous clusters.

Present multimedial retrieval systems have no, or fairly simple, graphical user interfaces. *VoiceGraph* is a graphical interface developed for audio and multimedia retrieval (Slaughter *et al.*, 1998). The interface was designed to facilitate rapid browsing of spoken documents. Another, web-based, graphical interface was developed for the *Taiscéalaí* radio broadcast retrieval system (Smeaton *et al.*, 1998). In *Taiscéalaí*, the retrieved documents are presented as time series of window scores. The users are allowed to select parts of a broadcast. The selections are delivered from the archive and transformed to a RealAudio file which is played to the user.

3 System Design

Most multimedial retrieval systems are modifications of existing text-based IR systems, disregarding the particular problems caused by the new media types. However, in order to manage the problems presented in Section 2 we need to test new solutions and experiment with several different algorithms. So instead of using existing text-based platforms we decided to design and develop a new flexible IR system. The requirements for a retrieval toolkit can be formulated as:

- A *modular* design is desirable: it should be easy and fast to make small changes to specific parts of the toolkit without reimplementing the whole system.
- The system must be *extensible* and *open*, since we want to be able to experiment with new algorithms and new media types. Thus, adding new modules has to be fairly straight-forward.
- The toolkit has to run *dynamically* and *flexibly*, allowing for quick adaptations to new requirements at any time by connecting/disconnecting modules.

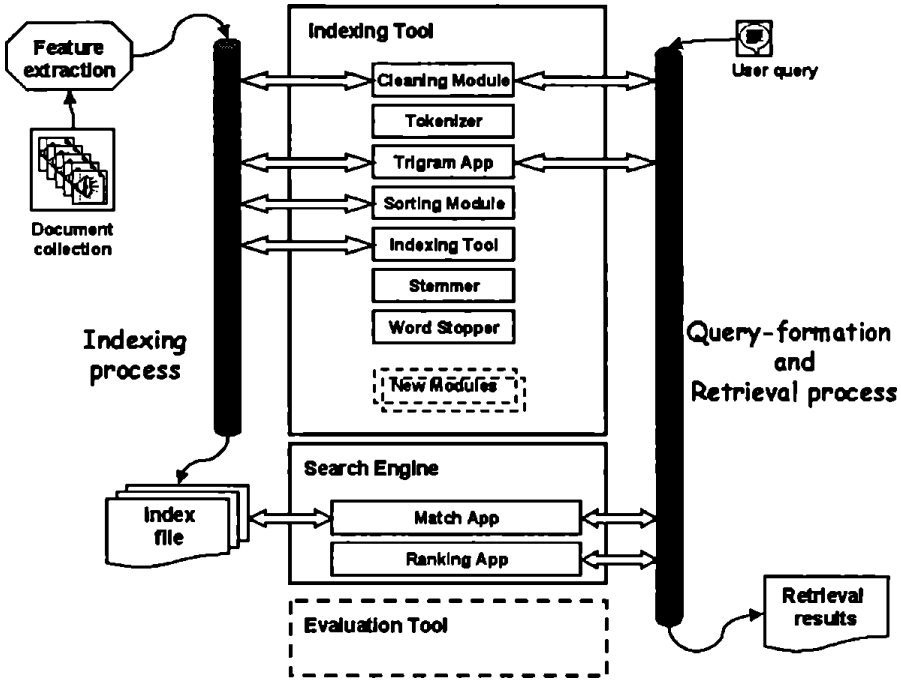


Figure 2: The design of SIREN

- Another desirable quality is *transparency*: we might want to examine the data flow under certain conditions.
- Due to the distributed nature of digital libraries, *interoperability* within a library architecture is necessary. To achieve a high level of access and sharing, we need a general framework for handling information across various domains such as different communities and different types of information objects.
- *Portability* is not necessarily a major requirement, but in itself a desirable quality of software systems.

We chose the object-oriented Java environment in order to address the portability issue.[1] When implementing the search engine, we used the built-in `Thread` class features, allowing us the option of adapting the system to the use of parallel search threads for searching multiple index files. This is essential when several media types are searched simultaneously.

The two major components of our IR toolkit are the indexing tool and the search engine as shown in Figure 2. (A third component, an evaluation tool, is planned.) The IR solutions and algorithms are based on well-founded results: We use a common inverted-file structure for storing indexing and posting information. The index-term weighting scheme is the *tf*idf* formula. The search engine implements the vector-space model and the cosine metric matching function. Thus, the novelty of the approach lies not in the

[1]Certainly, the choice of programming language is crucial for system performance, and Java is not well-known for its speed qualities. However, it is still under development, promising better speed properties. Also, modular system design coupled with the Java environment makes reimplementation of vital components in more time-effective languages tractable.

algorithms used, but in the way these algorithms are implemented in the system, and in the system design which allows for fast reimplementation and integration of new modules.

3.1 The modules

The main tools are built up of several small, independent modules, each of them taking care of one well-defined atomic task. The main advantage of this modular design is the possibility of reusing the modules in both the indexing and the query-formation process.

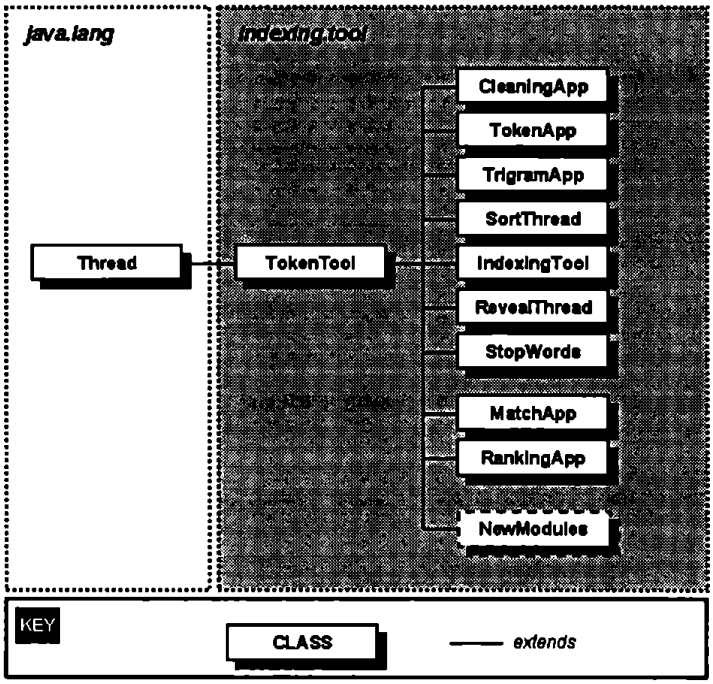


Figure 3: The SIREN class structure

The modules are implemented as independent Java classes as shown in Figure 3. Some examples of the tasks performed by the indexing tool: parse a document into a word-based token stream; parse a document into a trigram-based token stream; remove stop-words from the stream; perform morphological analysis; sort the stream; index the token stream.

As indicated in Figure 2, the modules are connected together in a pipelined architecture, using the Java built-ins `PipedReader` and `PipedWriter` for input and output, respectively. In this way, pipe syncronisation is taken care off automatically. The connections between the modules are dynamic, so that modules can be invoked "on the fly". So could, for example, the indexing tool be called with the following command sequence

```
java Siren clean token sort index words.txt
```

which would lead to the file `words.txt` being passed through the cleaning, tokenisation, sorting, and indexing modules. Modules can even behave in a mutually interchangeable way, if they perform similar tasks. Thus, it is possible to change the indexing feature

from words to trigrams by just changing the `token` command to `trigram` when invoking the tool as above.

A specific parser was implemented for reading the command lines. This tool scans the input stream and activates the corresponding modules. The order in which the commands are written on the line is critical. The system runs the modules in the same order as the command line indicates. Running modules in a faulty order does not result in system failure, but probably makes no sense in the overall indexing or retrieval process. In the worst case, the index file could be updated with wrong index terms. The parser accepts two different types of commands. The simple, one-word command calls one specific module which is operating on the data stream. The second type of command needs an additional argument which is assumed to be a file name. That file may be used for either input or output, depending on the specific module's design and purpose.

An advantage of the piped architecture is that the need of using and saving intermediate files is eliminated. Implementing and integrating new modules is fairly straight-forward. A new module is implemented as a new Java class and can take advantage of the built-in token language. To integrate a new component, only new method calls have to be declared, as long as the module supports `PipedReader` and `PipedWriter`.

The main requirement on the modules is to extend the `TokenTool` class described in the next section. The modules must also implement a `run` method. In addition, each module needs to have a corresponding calling method. These methods have to be declared just once. The system keeps track of the available methods through the Java built-in `Reflect` class.

The system is *open*, thus, the pipes are open at both ends and adding new modules with specific preprocessing tasks, such as prosodic analysis is possible, as is also connecting the pipe directly to a speech recogniser. Appending new modules for relevance feedback or retrieval evaluation is also feasible.

3.2 The token language

The modules are communicating through a common token stream, a solution slightly inspired by University of Glasgow's SIRE system (Sanderson and Crestani, 1998). The token stream consists of simple ASCII text. The modules are operating on the stream by modifying the tokens, by adding new tokens, or by removing some of them. Each token holds one term and a number of additional attributes associated to it. The sorting module assumes that the first component in every token is the indexing term. The modules are not sensible to the order of the other attributes. A small example of a token stream is given in Table 1.

The first column contains the indexing terms. In this example, a list of sorted trigrams. The second attribute is an internal tag and carries information about the type of the term. Here the `tg` tag indicates that the terms are in trigram-form. The typing is not a strong typing. Any character combination is allowed except for tab and newline. The number following indicates the term's position in the original file (byte offset). The last component is the term's *tf* score in the document. Adding new attributes can be easily done by just appending the tab separator and the desired piece of information. In the above example, we might include some information about the index-terms' weight scores.

Table 1: An example token stream

term	tag	offset	*tf*
tha	tg	1034	2
the	tg	1010	19
thi	tg	950	1
tic	tg	1066	3
tig	tg	927	1
tin	tg	323	2
tio	tg	1006	6
tis	tg	274	1
tit	tg	136	1
tl'	tg	572	2
tly	tg	369	2
tme	tg	82	1
toc	tg	293	1

The tokens are processed as they are by most of the modules. Converting the stream to an internal representation is needed in just a few cases, for example, in the sorting module. This is a nice quality since it enhances execution performance. By using pipes as the linking method, it is possible to inspect the data flow between two different modules by calling a specially designed `RevealThread` module which saves or prints a copy of the data flow.

Due to the ASCII nature of the token stream, the resulting data can be manipulated as plain text. However, the underlying data representation is hidden from the modules in a object-oriented manner. Specially built `TokenTool` class methods provide a better abstraction level in manipulating the attributes connected to the tokens. Thus, the modules are separated from the data representation and rather use general IR concepts, such as index-term, weight, frequencies, etc. This solution allows a more general implementation of the IR modules and operations, so that data manipulation is independent of the addressed media type. In addition, the system implementation also allows for an eventual redesign and reimplementation of the data representation without major impacts on the system design and on the system components.

4 Conclusions / Future Work

We have described SIREN, a flexible, modular information retrieval system designed to allow for different approaches to addressing the particular problems which are encountered when attempting to access non-text documents. The toolkit is already functional and useful for text and spoken document retrieval. New modules with specific tasks will soon be added, such as stylistic analysis and name recognition.

References

Abberley, D., Renals, S., and Cook, G. 1998. Retrieval of Broadcast News Documents with the THISL System. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington. IEEE.

Bretan, I., Dewe, J., Hallberg, A., Wolkert, N., and Karlgren, J. 1998. Web-Specific Genre Visualization. In *Proc. 3rd World Conference on the WWW and Internet*, Orlando, Florida. AACE.

Bub, T., Wahlster, W., and Waibel, A. 1997. Verbmobil: The Combination of Deep and Shallow Processing for Spontaneous Speech Translation. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 71–74, München, Germany. IEEE.

Crestani, F. and Sanderson, M. 1997. Retrieval of Spoken Documents: First Experiences. Technical report, Dept. of Computing Science, University of Glasgow, Scotland.

Delacourt, P., Kryze, D., and Wellekens, C. J. 1999. Speaker-based Segmentation for Audio Data Indexing. In Robinson, T. and Renals, S., eds., *Proceedings of the Workshop on Accessing Information in Spoken Audio*, pp. 78–83, Cambridge, England. ESCA.

Falk, J. 1997. Pauses in Synthesized Speech: Automatic Prediction of Silent Intervals in Swedish. Master of Art Thesis, Dept. of Linguistics, Göteborg University, Sweden.

Gambäck, B., Eineborg, M., Eriksson, M., Ekholm, B., Lyberg, B., and Svensson, T. 1995. A Language Interface to a Polyphone-Based Speech Synthesizer. In *Proc. 4th European Conference on Speech Communication and Technology*, volume 2, pp. 1219–1222, Madrid, Spain. ESCA.

Garofolo, J. S., Voorhees, E. M., Stanford, V., and Jones, K. S. 1997. TREC–6 1997 Spoken Document Retrieval Track Overview and Results. In Voorhees, E. M. and Harman, D. K., eds., *Proceedings of the 6th Text Retrieval Conference*, pp. 83–91, Gaithersburg, Maryland. National Institute of Standards and Technology.

Glavitsch, U. and Schäuble, P. 1992. A System for Retrieving Speech Documents. In Belkin, N., Ingwersen, P., and Mark Pejtersen, A.-L., eds., *Proc. 15th International Conference on Research and Development in Information Retrieval*, pp. 168–176, København, Denmark. ACM SIGIR.

Glavitsch, U., Schäuble, P., and Wechsler, M. 1994. Metadata for Integrating Speech Documents in a Text Retrieval System. *SIGMOD Record*, **23**(4):57–63.

Hakkani-Tür, D., Tür, G., Stolcke, A., and Shriberg, E. 1999. Combining Words and Prosody for Information Extraction from Speech. In Prószéky, G., Németh, G., and Mándli, J., eds., *Proc. 6th European Conference on Speech Communication and Technology*, volume 5, pp. 1991–1994, Budapest, Hungary. ESCA.

Hauptmann, A. G. 1995. Speech Recognition in the Informedia™ Digital Library: Uses and Limitations. In *Proc. 7th International Conference on Tools with AI*, Washington, DC. IEEE.

Hess, W., Batliner, A., Kiessling, A., Kompe, R., Nöth, E., Petzold, A., Reyelt, M., and Strom, V. 1996. Prosodic Modules for Speech Recognition and Understanding in Verbmobil. In Sagisaka, Y. *et al.*, eds., *Computing Prosody: Approaches to a Computational Analysis and Modelling of Prosody of Spontaneous Speech*, pp. 363–384. Springer, New York, New York.

James, D. A. 1995. *The Application of Classical Information Retrieval Techniques to Spoken Documents.* Doctor of Philosophy Thesis, Downing College, Engineering Dept., University of Cambridge, England.

Jones, G. J. F., Foote, J. T., Sparck Jones, K., and Young, S. J. 1996. Retrieving Spoken Documents by Combining Multiple Index Sources. In Frei, H. *et al.*, eds., *Proc. 19th International Conference on Research and Development in Information Retrieval*, pp. 30–38, Zürich, Switzerland. ACM SIGIR.

Kokkinakis, G., Fakotakis, N., and Dermatas, E., eds.. 1997. *Proc. 5th European Conference on Speech Communication and Technology*, Rhodes, Greece. ESCA.

Lieske, C., Bos, J., Gambäck, B., Emele, M., and Rupp, C. 1997. Giving Prosody a Meaning. In Kokkinakis *et al.*, eds. 1997, pp. 1431–1434.

Ng, K. 1998. Towards Robust Methods for Spoken Document Retrieval. In *Proc. 5th International Conference on Spoken Language Processing*, Sydney, Australia.

Ng, K. and Zue, V. 1997. Subword Unit Representations for Spoken Document Retrieval. In Kokkinakis *et al.*, eds. 1997, pp. 1607–1610.

Nikolaou, C. and Stephanidis, C., eds. 1998. *Proc. 2nd European Conference on Research and Advanced Technology for Digital Libraries*, Heraklion, Greece.

Robinson, T., Hochberg, M., and Renals, S. 1996. The Use of Recurrent Networks in Continuous Speech Recognition. In Lee, C.-H. and Soong, F. K., eds., *Advanced Topics in Automatic Speech and Speaker Recognition*, chapter 7. Kluwer, Dordrecht, Holland.

Sanderson, M. and Crestani, F. 1998. Mixing and Merging for Spoken Document Retrieval. In Nikolaou and Stephanidis, eds. 1998, pp. 397–407.

Schäuble, P. and Wechsler, M. 1995. First Experiences with a System for Content Based Retrieval of Information from Speech Recordings. In *Proc. IJCAI Workshop on Intelligent Multimedia Information Retrieval.*

Slaughter, L., Oard, D., Warnick, V., Harding, J., and Wilkerson, G. 1998. A Graphical Interface for Speech-Based Retrieval. In *Proc. 3rd Digital Library Conference*, Philadelphia, Pennsylvania. ACM.

Smeaton, A., Morony, M., Quinn, G., and Scaife, R. 1998. Taiscéalaí: Information Retrieval from an Archive of Spoken Radio News. In Nikolaou and Stephanidis, eds. 1998, pp. 429–442.

Stolcke, A., Shriberg, E., Hakkani-Tür, D., Tür, G., Rivlin, Z., and Sönmez, K. 1999. Combining Words and Speech Prosody for Automatic Topic Segmentation. In *Proc. Broadcast News Workshop*, Herndon, Virginia. DARPA.

Wechsler, M. 1995. Eine neue Indexierungsmethode für Information Retrieval auf Audiodokumente. In *Proc. Hypertext-Information Retrieval-Multimedia*, pp. 117–128.

Statistics and Phonotactical Rules in Finding OCR Errors

Stina Nylander

Uppsala University & SICS

stina@stp.ling.uu.se

Abstract

This report describes two experiments in finding errors in optically scanned Swedish without lexicon. First, statistics were used to find unexpectedly frequent trigrams and correction rules were created. Second, Bengt Sigurds model of Swedish phonotax was used to detect words with phonotactically illegal beginning or end.

The phonotax did not perform as well as the statictic rules did on their training material, but outscored them by far on new text.

A correction tool was created with the phonotax as means of error detection. The tool displays every occurrence of an error string at the same time and gives the user the possibility to give different corrections to each occurrence.

This work shows that it is possible to find errors in optically scanned text without relying on a lexicon, and that word structure can provide useful information to the correction process.

1. Introduction

Optical character recognition (OCR) is a technique for moving text resources from paper medium to electronic form, something that is often needed in our computerised society. Companies and authorities want to make old material machine readable or searchable. Unfortunately, it does not get us all the way. With good paper originals, OCR can achieve 99% of the characters correctly recognised but the result will still contain in average one error word per 20 words which means 5% incorrect words or about one error per sentence (Kukich, 1992). Depending on the application of the optically scanned text, large post processing efforts can be necessary. Since OCR is often used to move large amounts of text to electronic form, the proofreading is a task both demanding and dull. This makes the need for good tools of spell checking and correction large and urgent.

Most spell checkers and OCR post processing systems are lexicon based. A lexicon of reasonable size is used to match against the text, and any word token not in the text is presented as a possible error. Probability scores or similarity measures are then used to generate correction suggestions.

I will concentrate on the error finding process and not try to generate correction suggestions. I want to find ways of proofreading text without relying on a lexicon. Instead I will try to define rules that identify character sequences that are unlikely to be correct word tokens. I made two experiments: using statistical methods to find unexpectedly frequent character sequences, and using phono- or graphotactical rules to find unlikely character combinations. Obviously these results can be generalised for all kinds of proofreading tasks: e.g. handwriting recognition or dictation tasks.

The work described in this report has been done within a Master thesis at the Language Engineering Programme at Uppsala University. The work has been carried out at SICS and was funded by the Digital Library project.

1.1 OCR errors

Many recognition errors are caused by graphical similarity : `arguinent` (argument), `teature` (feature), `rnean` (mean), `sernantics` (semantics), `systernet` (systemet), `textförstäelse` (textförståelse), `disambiqueras` (disambigueras).

Proofreading by hand is difficult. The graphical errors are by definition difficult to detect by ocular scanning through the text: the visual difference between *bodv* and *body* is very small. Other problems are print quality, font and the age of the original that sometimes produce errors that make it impossible to guess the original word like `appTijdo-tiL-s` (approaches.) or `Umt` (that).

Another group of errors that occurs in optical scanning of text is split errors; spaces are inserted in a word and produces a number of strings, many of them incorrect: `pronunc i at ion` (*pronunciation*), `i nt e` (*inte*), `öre I i gger` (*föreligger*).However, many or even most of these errors still produce string tokens that are unlikely or impossible words in the language under consideration.

1.2 Approaches to Error Correction

Most approaches to correction of scanning errors are lexicon based. A lexicon of reasonable size is used to match against the text, and any word token not in the lexicon is defined as incorrect. This leads to many false alarms, since a lexicon never can cover everything. Many correct words and proper names will be presented as errors by the system. To find real word errors -- i.e. errors that result in another correct word -- sequences of parts of speech are evaluated for likelihood of occurrence, and unlikely sequences are marked as possible errors.(Meknavin et al., 1998; Tong & Evans 1996; Huismann, 1999).

The research made shows that OCR post processing problems are highly language specific, Meknavin et al. show that one of the biggest problems working with Thai is to find the word boundaries (1998), while those working with English put the largest effort in spotting the errors (Tong & Evans, 1996; Golding & Shabes, 1996), or providing good correction suggestions (Takahashi et al., 1990). Lee et al. argues that the Korean writing system is syllable based and that recognition and error correction therefore should be syllable based rather than character based (1997), and Hogan points out that if you work with minority languages, in his case Haitian Creole, it is not likely that you even use OCR software developed for your language, which makes post processing even more necessary.

We want to find ways of proofreading text without relying on a lexicon by finding character sequences that are unlikely to be correct word tokens. We tried two experiments: using statistical methods to find unexpectedly frequent character sequences, and using phono- or graphotactical rules to find unlikely character combinations.

2. Statistics

The hypothesis is that differences in observed frequency between correct text and optically scanned text for a character n-gram would indicate that the n-gram in question was incorrectly recognized by the scanning process.

The NoDaLiDa conference proceedings were selected as experimental material. The proceedings contain both correct text as provided by the author in machine readable form and optically scanned text. Two optically scanned papers were handcorrected to obtain testing material: *How Close Can We Get to the Ideal of Simple Transfer in Multilingual Machine Translation (MT)?* (Andersen, 1989) of about 2500 words, henceforth NODA89-09, and the first part of *A self-extending lexicon: description of a word learning program* (Ejerhed & Bromley, 1985) of about 1800 words, henceforth NODA85-06. The statistical experiment was made on English text material, since we do not have enough Swedish text provided in machine readable form to establish frequencies for the correct text.

Since it is necessary to be able to count the number of errors automatically in studies like this, proofreading and error counting by hand is simply too costly in time, a simple error measure was defined for the experiment: the number of errors is the difference in number of word tokens between the optically scanned text and the corrected text plus the number of strings that only appeared in the optically scanned text. The real word errors would not be included in the resulting number of errors and split errors would be counted as the number of parts the original words was split into.

The n-gram frequencies of the corrected text and the optically scanned text were compared and n-grams that showed large frequency differences between text versions were displayed to the editor, together with a concordance of all the occurrences of the n-gram. This allowed the editor to formulate a correction rule for the n-gram under consideration.

Two sets of rules were formulated for each article, one with rules that replaced a character trigram with another string of optional length, the other with rules that replaced a string of optional length with another. The rules that rewrite trigrams were generated with the support of a graphical tool that generated a list of suspect trigrams, for each trigram showed a concordance of all the occurrences of the trigram, and, with a correction given by the user, could generate a correction rule. The rules that rewrites longer strings were generate by hand.

The rules were then used to correct both the article that had been used to generate the rules and the other, to see if the rules were useful in another context than the one they had been generated in.

The number of errors were counted by means of a perl program before and after correction to estimate the performance of the rules. The number of errors generated in the correction process were counted separately to keep track of over correction. As generated errors were considered strings that appeared only in the version corrected with the rules, neither in the correct version nor in the optically scanned version. Over corrections that result in correct words (or already present errors) will thus not be counted.

The tests described above show that while rules based on observed frequencies of character sequences do provide a noticeable improvement on the training material and presumably will be useful for proofreading a given text, they are too specific for use on other material. When it comes to the trigram rules the problem could be that the trigram as context is too small and that many errors, and even more corrections, affect more than three characters, but even if the rules that treated longer strings worked a little better on unknown errors, they were much too text specific too. All the rules deal more or less with a given error in a given word, which is even more true for the rules that rewrote strings of optional length: the longer string that is rewritten, the less generic is the rule. This approach needs a very large text material to generate the rules from and a huge number of rules to be of any significant use in correcting new texts.

3. Graphotactical Rules

Instead of trying to find potential error strings by computing frequency data and comparing correct text with optically scanned text, we used Bengt Sigurd's model of Swedish phonotactics (1965). The model was adapted to graphemes, i.e. the phoneme /ʃ/ was replaced by all the different Swedish spellings (sk, sj, stj, skj etc.) and the same for the phonemes /j/ and /ç/. The SUC corpus (Källgren, 1990) was used to check which vowels that followed the different spellings of these phonemes, *stj* being followed only by *ä*, *ch* by *a,o,e,i*, etc. This added 63 initial consonant combinations to Sigurds 55. The legal initial vowel clusters were listed empirically by extracting all words from SUC that started or ended in a sequence of vowels and added 22 initial sequences to the model. Altogether this gives around 530 different initial consonant sequences + vowel or initial vowel sequences.

Sigurd has 102 primary final consonant combinations. To this has been added 11 combinations to cover the Swedish final doubling of consonants after long vowel, *-x* and *-xt* as final consonant clusters since *x* is realized as two phonemes, /ks/, and is thus as letter not a part of Sigurds description (Benny Brodda even says that the letter *x* has no place in a phonological description of Swedish (Brodda, 1979)). The legal final vowel clusters were listed empirically by extracting all words from SUC that ended in a sequence of vowels. With these adaptions made, the model could be used to find strings beginning or ending with illegal consonant clusters.

In addition rules were added to find strings with consonants only, strings with mixed alphanumeric characters, strings with mixed case and strings with punctuation characters in non-final positions. With these adaptations made, the model could be used to find strings beginning or ending with illegal consonant clusters.

Breaches of the graphotactical rules were extracted from the text and marked as potential errors. The user was displayed the list of possible errors, asked to suggest a correction, and to mark which occurrences should be corrected.

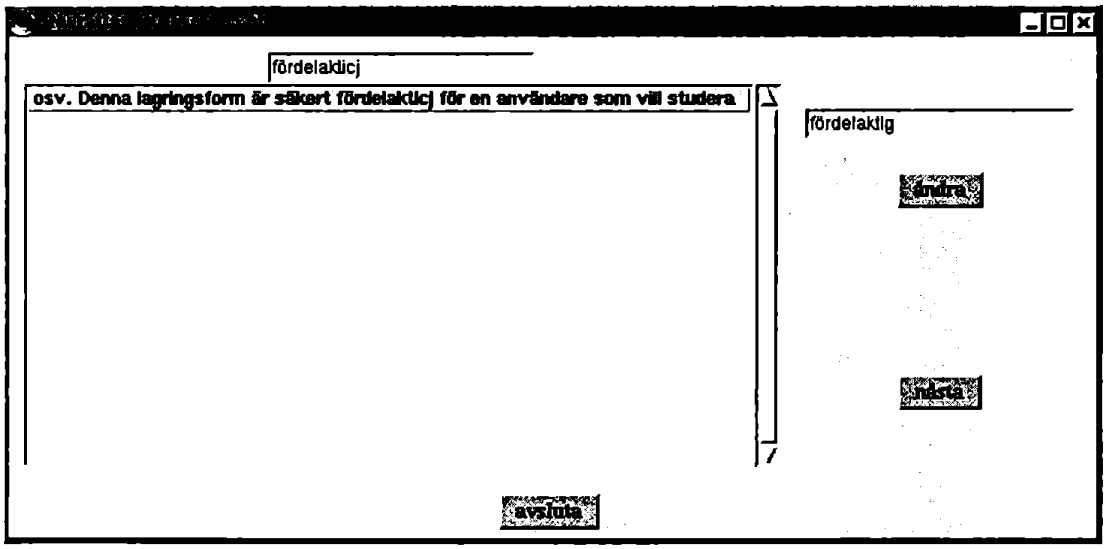
A corpus of optically scanned Swedish text containing 71 000 words was scanned for graphotactical clashes. We found 2495 words with possible errors (822 words with illegal prefixes, 737 with illegal suffixes, 336 words containing punctuation marks or other special characters and 600 words that mixed letters and digits or upper and lower case characters). Of these, many are abbreviations, foreign words, or correct Swedish words with unusual spelling.

State-of-the-art OCR systems give a result of up to 99% correct character recognition, which gives on average one error per 20 words (Kukich, 1992). One error per 20 words would for our corpus give 3550 errors, and while we found 2495 possible errors, about 370 of them were abbreviations (which could easily be filtered out), about 75 were correct non-compound words, and about 100 were acceptable alphanumeric combinations. This leaves us about 1900 likely errors -- a precision of around 75% at a recall of more than 50% if the error estimate holds! And the graphotactic rules can be improved -- at the moment they only deal with initial and final clusters.

The rules were also tested on a single article, *Inte bara idiom* containing 2200 words (Allén, 1983), corrected by hand. The article contained 89 errors, 4% of the total number of words, of which 24 were real word errors and 1 was a split error, both error types that the rules can not handle. The rules presented 42 possible errors of which 19 were errors, 1 was a correct Swedish word, 14 were correct foreign words and 8 were abbreviations. This means that, when the abbreviations have been filtered out, the rules managed to find 29% of the non word errors and only presented one correct Swedish as an error.

4. Implementation

The implementation of the graphotactical experiment described above is a correction tool for Swedish (html) text written in Perl/Tk. The program detects non word errors with the phonotactical rules described above and enables the user to proofread a text in a non linear way. All occurrences for each possible error are displayed to the user at the same time, with a small context. This gives the user the possibility to decide if all the occurrences really are to be corrected, or if one occurrence is correct (maybe an unusual abbreviation or an acronym). Each occurrence can then be given a different correction if



necessary.

4.1 Possible Improvements

At the moment there is no possibility of undoing a made correction, since the program does not keep track of where the correction is made in the text. The program can not find the position in the text again, and thus can not undo the correction. Another consequence of this is that the user can do only one correction per occurrence.

It is not possible to go back to the previous error word and to see the concordance over that word again. The program does not keep the error words and can thus not go back and reconstruct the concordance.

When correcting an error the user should be able to change the scope of the error. If the program presents *översättn* as a possible error and the context looks like *översättn ing*, the user should be able to mark *översättn ing* as the error string and replace it with the correct string *översättning* without space. At the moment the user can not change the scope of the error string, thus split errors cannot be corrected even when observed by the user.

5. Discussion

The above experiments show that it is possible to find errors in text without relying on a lexicon, and without the large numbers of false alarms we have learnt to expect from such systems. And -- which should not be surprising to those of us who are linguists! -- it is also clear that knowledge of the structure of words improves the results. The phonotactical rules might not reach the same recall as a lexical error finding approach, strings that do not violate the Swedish phonotactics might still be non words, for example *sernantik* (semantik) and *systernet* (systemet). The precision of this method although, will spare the user many of the false alarms and still clean up the text from a substantial part of the recognition errors.

References

Allén, S. 1983. Inte bara idiom. In Proceedings of the 4th Nordic Conference on Computational Linguistics. Uppsala.

Andersen, P. 1989. How Close Can We Get to the Ideal of Simple Transfer in Multilingual Machine Translation (MT)? In Proceedings of the 7th Nordic Conference on Computational Linguistics. Reykjavik.

Ejerhed, E. & Bromley, H. 1985. A self-extending lexicon: description of a word learning program. In Proceedings of he 5th Nordic Conference on Computational Linguistics. Helsinki.

Hogan, C. 1999. OCR for Minority Languages. In Proceedings of the 1999 Symposium on Document Image Understanding Technology. Annapolis, Maryland.

Huismann, G. 1999. OCR Post Processing. Groningen University. Groningen.

Kukich, K. 1992. Techniques for Automatically Correcting Words in Text. In ACM Computing Surveys, Vol. 24, No. 4, 377-439.

Källgren, G. 1990: ``The first million is hardest to get'': Building a Large Tagged Corpus as Automatically as Possible. In Proceedings of COLING 90. Helsinki.

Lee, G., Lee, J-H., Yoo, J. 1997. Multi-level post processing for Korean character recognition using morphological analysis and linguistic evaluation. Pattern Recognition 30(8): 1347 - 1360.

Meknavin, S., Kijsirikul, B., Chotimonkol, A. Nuttee, C. 1998. Combining Trigram and Winnow in Thai OCR Error Correction. In Proceedings of COLING 1998. Montréal.

Sigurd, B. 1965. *Phonotactic Structures in Swedish.* Lund University. Lund.

Takahashi, H., Itoh, N., Amano, T. & Yamashita, A. 1990. A Spelling Correction Method and Its Application to an OCR System. Pattern Recognition vol 23 3/4.

Tong, X. & Evans, D. 1996. A Statistical Approach to Automatic OCR Error Correction in Context.

An Information Retrieval System

with Cooperative Behavior

Paulo Quaresma and Irene Pimenta Rodrigues
pq@di.uevora.pt ipr@di.uevora.pt

Departamento de Informática
Universidade de Évora
7000 Évora
Portugal

CENTRIA/AI Center
Universidade Nova de Lisboa
2825 Monte da Caparica
Portugal

Abstract

In this paper we will present a system that is able to perform cooperative information retrieval actions over a text knowledge base. The knowledge base is composed by four levels: Interaction, Domain, Information Retrieval and Text. The interaction level is responsible for the dialogue management, including the inference of attitudes. The domain level is composed by rules encoding knowledge about the text domain. The information retrieval level includes knowledge about IR actions over sets of documents. The text level has knowledge about the words in each text.

Cooperation is achieved through two main strategies: 1) clustering the answer sets of documents accordingly with the domain and IR-level knowledge; 2) keeping the context of the interaction and inferring the user intentions.

1. Introduction

In this paper we present a cooperative information retrieval system in the law domain. The information retrieval system is composed by a text knowledge base built from the Portuguese Attorney General documents.

As an example, the following dialogue demonstrates the major features of our system (the example will be explained in more detail in the last section).

User - U1: Documents about pensions for relevant services?
System - U2: Pensions that were given or refused?
User - U3: Both.
System - U4: Pensions for militaries or civilians?
User - U5: Civilians
System - U6: There are 129 documents, listed in the right frame.
User - U7: where there was a drowning?
System - U8: Since there are 45 documents of the 129 above that match the concept, I've listed them in the usual frame.
User - U9: No, I mean to look for all documents that match the concept.
System - U10: There are 347 documents listed in the right frame.
User – U11: Documents about drugs?

System – U12: I assume that you want a new query about drugs since there are no documents with the concept drugs and pensions for relevant services for civilians. Do you want to see documents about the use of drugs while driving?

User: - U13 yes.

...

In order to handle dialogues similar with this one, our system needs to be able to keep the context of the interaction and to cooperatively supply suggestions for further refinement of the user queries. The refinement process is based on the domain knowledge and the ability to compute clusters of documents associating a keyword (from a juridical thesaurus with 6000 expressions) to each cluster.

In order to perform a cooperative interaction with the user, the system should be able:

- To infer what are the user intentions with the queries. For instance, when a user asks for documents with a particular keyword, he may be interested in documents that do not have that exact keyword and he may not be interested in all documents with that keyword.
- To supply pertinent answers or questions as a reply to a user question. The system must supply some information on the set of documents selected by the user query in order to help him in the refinement of his query.

As a consequence our system needs:

- To record the user interactions with the system. User interactions will provide the context of sentences (questions and answers), allowing the system to solve some discourse phenomena such as anaphoras and ellipses.
- To obtain new partitions (clusters) of the set of documents that the user selected with his query(ies). The clustering process should be based on the text knowledge representation.

In the next section we will describe the text knowledge base. Then, in section 3 and section 4 the interaction structure and the inference of attitudes will be described. In section 5, the clustering process will also be described. In section 6, an example of a cooperative session will be presented. Finally, in section 7, conclusions and future work will be presented.

2. Knowledge Base

The knowledge base is composed by four levels: Interaction, Domain, Information Retrieval and Text.

1. The interaction level is responsible for the dialogue management. This includes the ability of the system to infer user intentions and attitudes and the ability to represent the dialogue sentences in a dialogue structure in order to obtain the semantic representation of the dialogue;
2. The domain level includes knowledge about the text domain and it has rules encoding that knowledge. For instance, in the legal domain it is necessary to represent under which conditions a pension for relevant services may be given to someone; those pensions are usually attributed to militaries or to civilians such as firemen, doctors, and nurses;

3. The Information Retrieval Level includes knowledge about what we should expect to find in texts about a subject; for instance that in texts about pensions for relevant services, the pension may be attributed or refused;
4. The Text Level has knowledge about the words and sequence of words that are in each text of the knowledge base. This level is based on SINO, a text search engine with inverted files from the AustLII Institute [Greenleaf et al. 1997] that was extended to the Portuguese language. The extended SINO is able to access a 900,000 Portuguese lexicon and it is able to handle morphological information (verbal forms, plurals, etc.)

These four levels of knowledge are integrated via a dynamic logic-programming module, which is responsible for the management of the interaction with the users.

Dynamic logic programming [Alferes et al. 1998] defines how a logic program can be updated by other logic programs. In our approach, each event is represented by a logic program (composed only by facts), which is used to update the previous program and to obtain a new one. In fact, events are represented by an update situation and there is no need to explicitly represent time points. Inertia rules are also guaranteed by the dynamic logic programming semantics.

3. Interaction structure

The system builds the interaction structure to record both user and system questions and answers. This structure is used to compute the meaning of an user query and to allow the user to return to a previous point of the interaction and to build a new branch from there.

The Interaction Structure (IS) is made of segments that group sets of interactions. At present we are able to deal with 3 different kinds of segments:

- Basic --- has 2 arguments: Speaker;Action Representation
- New --- has 2 arguments: Interaction Structure; Interaction Structure. The new IS inherits its attributes from the second argument. Ex: New([],basic(User,Q1))
- Specify --- has 2 arguments: Interaction Structure; Interaction Structure. The new IS inherits its attributes from both interaction structures. Ex: Specify(Basic(User,Q1), Basic(System, Q2))

3.1. Rules to build the interaction structure

Given an action A1 from an agent A, the update of the new action is:

action(basic(A,A1)).

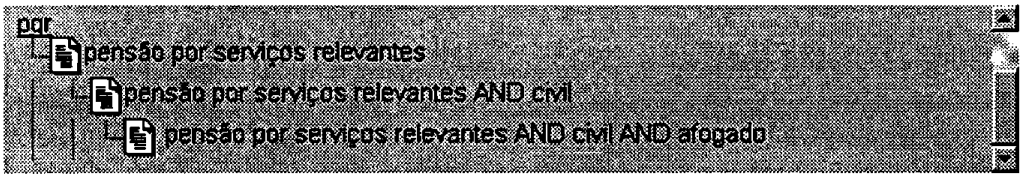
This fact gives rise to the update of the new Interaction Structure according to the above rules:

```
is(specify(Ois,Is)) <- is(Ois)/past, action(Is)/now,
                       bel(system,specify(Ois,Is)/now.

is(new(Ois,Is)) <- is(Ois)/past, action(Is)/now,
                   bel(system,new(Ois,Is))/now.
```

These two rules encode that the new interaction structure is a structure that includes the semantics of the new action and that the system is able to infer that at this point of the interaction is believable.
The conditions for this system belief can be defined in many different ways, but our system normally beliefs that users intend to specify previous actions.

```
bel(system, specify(Ois,Is)) <- not neg bel(system, specify(Ois,Is)).
bel(system, new(Ois,Is)) <- bel(system, incompatible(Ois,Is)).

neg bel(system, specify(Ois,Is)) <- bel(system, new(Ois,Is)).
neg bel(system, new(Ois,Is)) <- bel(system, specify(Ois,Is)).
```

Where "not" means default negation and "neg" means explicit negation.



As it is shown, the system displays a graphic representation of the interaction in order to help the user to keep in mind the interaction context. Moreover, it allows the user to select a node in the tree for defining the context of his next query. This feature has shown to be very useful since our users use it very frequently.

4. Inference of user intentions

In order to be collaborative our system needs to model user attitudes (intentions and beliefs). This task is also achieved through the use of logic programming framework rules and the dynamic LP semantics [Pereira and Quaresma 1998].

The system mental state is represented by an extended logic program that can be decomposed in several modules (see [Quaresma and Lopes 1995] for a complete description of these modules):

- Description of the effects and the pre-conditions of the speech acts in terms of beliefs and intentions;
- Definition of behaviour rules that define how the attitudes are related and how they are transferred between the users and the system (cooperatively).

For instance, the rule which describes the effect of an inform and a request speech act from the point of view of the receptor (assuming cooperative agents) is:

bel(A,bel(B,P)) <- inform(B,A,P)/before.
bel(A,int(B,Action)) <- request(B,A,Action)/before.

In order to represent collaborative behaviour it is necessary to model how information is transferred from the different agents:

bel(A,P) <- bel(A,bel(B,P))/now, (not bel(A,P))/before.
int(A,Action) <- bel(A,int(B,Action))/now, (not neg int(A,Action))/before.

These two rules allow beliefs and intentions to be transferred between agents if they are not inconsistent with the previous mental state.

After each event (for instance a user question) the agents' model (logic program) needs to be updated with the description of the event that occurred. The act will be used to update the logic program in order to obtain a new model. Using this new model it is possible to obtain the intentions of the system.

4.1. System reasoning steps

Any user act (utterance or other) will cause a system update that will give rise to following reasoning steps:

1. Update of the user act;
2. Update of the new interaction structure using the interaction structure rules and the updated act;
3. Update of system intentions that were inferred from the effects of the action rules;
4. Execution of the system intended actions.

4.2. Cooperative inference of user Goals

In order to infer the user goals the system uses two representation levels: Domain Knowledge and Information Retrieval knowledge.

The Domain level

This level is used to obtain the domain models that are consistent with the user query.

Example:

pension(X)

will give the models: {pension, military},{pension, civilian}, {pension}

They are computed assuming that we have the domain rules:

```
pension(X) <- military(X), action(X,A), behind_duty(A).
pension(X) <- civilian(X), action(X,A),
              save_life(Y,A), life_at_risk(X,A), X !=Y.
```

This knowledge level is built from the laws used in the texts. For instance the law describing the requisites to obtain a pension for relevant services can be encoded by the previous rules. These rules state that:

- A military may have a pension for relevant services if he has been the agent of an action, and that action was behind is duty.
- A civilian may have a pension for relevant services if he has been the agent of an action that saves someone life and he has his live at risk.

These models are used to ask the user to refine his query.

Information Retrieval Level

This level of knowledge is built with rules that can be obtained by processing the text documents looking for keywords that give rise to disjoint sets of documents. By now we obtain these rules using a thesaurus with keywords for text juridical classification.

Example of rules:

```
pension(X) <- pension_given(X).
pension(X) <- pension_refused(X).

false <- pension_refused(X), pension_given(X)
```

These rules state that a document with the concept pension either mentions the concept attributed or rejected.

As it was explained in the previous sub-section, the system computes the models that are consistent with the user query and with the interaction structure. Then, if there are more than one possible model, the system asks the user to refine his query and to refine it.

5. Clustering

A service that an information retrieval system always provide is the organisation of retrieval results. SINO can also rank the output according to estimated relevance values. However, if there are large document groups with similar rankings, it is a good option to build clusters and to label the groups with some relevant keywords.

Clustering helps users by showing them some kind of pattern in the distribution and it allows the user to include or exclude sets of documents from further searches.

We use the already existing juridical classification of the Portuguese Attorney General documents to characterize each cluster and to inform the user on possible further choices for refining his query.
Our clustering method is based on the Scatter/Gather browsing paradigm that clusters documents into topically-coherent groups. It is able to present descriptive textual summaries that are build with topical terms that characterise the clusters. The clustering and reclustering can be done on-the-fly, so that different topics are seen depending on the subcollection clustered.

6. Example

The following dialogue is an example of a user session in our system, which demonstrates its major features.

User - U1: Documents about pensions for relevant services?
System - U2: Pensions that were given or refused?
User - U3: Both.
System - U4: Pensions for militaries or civilians?
User - U5: Civilians
System - U6: There are 129 documents, listed in the right frame.
User - U7: where there was a drowning?
System - U8: Since there are 45 documents of the 129 above that match the concept, I've listed them in the usual frame.
User - U9: No, I mean to look for all documents that match the concept.
System - U10: There are 347 documents listed in the right frame.
User – U11: Documents about drugs?
System – U12: I assume that you want a new query about drugs since there are no documents with the concept drugs and pensions for relevant services for civilians. Do you want to see documents about the use of drugs while driving?
User: - U13 yes.
...

Given a user utterance, such as U1, the system is able to cooperatively interact with user in order to refine its query.

The system reply to U1 will be U2. This reply is achieved by recognizing that this query can be refined since the texts that mention pensions can be divided into two disjoint sets, one where pensions were given and another one were pension were refused. This kind of knowledge is encoded in what we have called the Information Retrieval level.

After the user answer (that could be: given, rejected or both), by using knowledge of the Domain level the system will generate question U4. This is achieved by knowing that pensions by relevant service have different conditions when there is a military or a

civilian. This is juridical knowledge independent of the texts present in the text base and represented in the domain level.

As we described in previous sections, the system is also able to decide if the user intends to continue its previous query (its utterance is to be interpreted in the context of the previous dialogue) or to open a new query (a new interrogation context).

If, after U1 the user asks U7, the system will be able to decide that the user intends to look for text where there are a pension and a drowning. But if the user utters U11 instead of U7 the system will conclude that the user intends to open a new interrogation context.

This is achieved by using the Textual level that encodes knowledge about the texts words and expressions (concepts). Using our retrieval information system SINO, it is possible to see that are some texts where the concepts *pension and drowning* appears but no texts where the concepts *pension and drugs* appears. This is what the user expects the system behave in most cases. When this is not the case the user may clarify its query in order oblige the system to behave differently. For instance after U7 the system will reply U8 and the user may reply U9.

U9 will be understood by the system as a user clarification and it will forget the semantic content of sentences U1-U8 by opening a new context with U9. In order to interpret the sentence U9 in particular to solve the nominal anaphora *the concept*, the dialogue structure of sentences U1-U8 will be used.

7. Conclusions and Future Work

The Information Retrieval system presented in this paper is implemented and it has a public access from the WWWeb.
A preliminar evaluation was done by taking into account the system logs and user comments. By analyzing the system logs we obtained that:

- Most queries (90%) are done using the multimodal interface. Most users do not use the natural language interface, they prefer to use choice menus, or to use free text queries (keywords with booleanconnections).
- The interaction context is frequently used by our users (on average twice on each session). The users use it in order to return to a previous interaction point.
- The system suggestions for query refinement are used in 90% of the cases.
- Most of the system suggestions (70%) are obtained using the information retrieval level.

Regarding the portability of our IR system into other domains, the main issues are:

- A robust natural language grammar enabling to obtain the speach act associated to a user multimodal act. (it may involve to add some vocabulary and some knowledge representation rules, mainly a domain thesaurus).
- A knowledge base modeling some domain knowledge.
- The computation on-the-fly of document clusters with a topical expression associated with each. This will be our main source of knowledge to compute the system sugestions for further refinement.

References

Allen, J. & Kautz, H & Pelavin, R. & Tenenberg, J. 1991. *Reasoning about Plans.* Morgan Kaufman Publishers, Inc..

Alferes, J. & Pereira, L. 1996. Reasoning with Logic Programming. *Lecture Notes in Artificial Intelligence*, 1111. Springer.

Alferes, J. & Leite, J. & Pereira, L. & Przymusinska, H. & Przymuzinski, T. 1998. Dynamic Logic Programming. *Proceedings of KR'98- Knowledge Representaion.*

Cohen, P. & Levesque, H. 1990. Intention is choice with commitment. *Artificial Intelligence*, 42(3).

Carberry, S. 1988. Modelling the user's plans and goals. *Computational Linguistics*, 14(3):23--37.

Greenleaf, G. & Mowbray, A. & King, G. 1997. Law on the net via AustLII - 14 M hypertext links can't be right? *Proceedings of Information Online and On Disk'97.*

Grosz, B. & Sidner, C. 1986. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175--204.

Kamp, H. & Reyle, U. 1993. *From Discourse to Logic: An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Dordrecht: D. Reidel.

Litman, D. & Allen, J. 1987. A plan recognition model for subdialogues in conversations. *Cognitive Science*, (11):163--200.

Lascarides, A. & Asher, N. 1991. Discourse relations and defeasible knowledge. *In Proceedings of the 29th Annual Meeting of ACL*, 55--62.

Pinto, J. & Reiter, R. 1993. Temporal reasoning in logic programming: A case for the situation calculus. *Proceedings of the 10th ICLP.* MIT Press.

Pereira, L. & Quaresma, P. 1998. Modeling Agent Interaction in Logic Programming. *In Proceedings of the 11th International Conference on Applications of Prolog.* Tokyo, Japan.

Quaresma, P. & Lopes, J. 1995. Unified logic programming approach to the abduction of plans and intentions in information-seeking dialogues. *Journal of Logic Programming*, (54).

Rodrigues, I. & Lopes, J. 1992. Discourse temporal structure. *In Proceedings of the COLING'92.*

Rodrigues I. & Lopes, J. 1993. Building the text temporal structure. *In Progress in Artificial Intelligence: 6th Portuguese Conference on AI.* Springer-Verlag.

Rodrigues, I. & Lopes, J. 1997. AI5, An Interval Algebra for the temporal relations conveyed by a text. *In Mathematical Linguistics II,* Eds Carlos Martin-Vide, John Benjamins.

Song, F. 1991. A Processing Model for Temporal Analysis and its Aplication to Plan Recognition. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada.

An evaluation of the Translation Corpus Aligner, with special reference to the language pair English-Portuguese

Diana Santos & Signe Oksefjell
SINTEF Telecom and Informatics & IBA, University of Oslo
Diana.Santos@informatics.sintef.no & Signe.Oksefjell@hia.no

Abstract

In this paper we describe the evaluation of a language-dependent aligner. We begin by introducing the alignment program, explaining why it would be interesting to evaluate it with particular emphasis on the language pair English-Portuguese. A short presentation of the corpus used to test the aligner is also given. We then describe three experiments that were performed in the evaluation process, presenting the results and discussing the methodology. The paper ends with a discussion of more general conclusions relative to an evaluation of this kind.

1. Introduction

Two criteria that are often employed in the evaluation of NLP programs are performance and usability. Another criterion, less frequently mentioned, is the adequacy of handling particular languages. The present study describes a set of experiments devised to perform such an evaluation.

Although researchers concerned with parallel corpus building and exploration will generally be happy to use a system available for their languages without evaluating it thoroughly, especially when the system is freely distributed – as is the case of the present system, the kind of work reported originates from two relevant concerns. The first one is about methodological aspects related to the development of NLP systems. The second concern is evaluation and comparison of products. In fact, there is a blatant lack of serious evaluation work of products and systems concerning the Portuguese language, which is a situation we have been trying to change in the project Computational Processing of Portuguese at SINTEF.[1]

The Translation Corpus Aligner (TCA) was developed in connection with the English-Norwegian Parallel Corpus (ENPC) project with the aim of automatically aligning English and Norwegian texts (see e.g. Hofland 1996, Hofland & Johansson 1998). Although the program was originally written for the language pair English-Norwegian, it has been further developed to handle other language pairs, including English-Portuguese. It includes a language-dependent component in the form of an anchor word list.

In the present paper we set out to evaluate the TCA for the language pair English-Portuguese. In particular, we want to

- investigate the effect of the anchor word list;
- compare the results of the program with and without the anchor word list;
- find out how much a proofreader has to check manually after alignment

In order to perform the evaluation, we used the English-Portuguese part of the ENPC, which currently includes 16 English texts, about 12,000 words each, that have translations into Portuguese.[2]

2. A short description of the TCA

The alignment program automatically matches original sentences with their translations. In the process of linking corresponding sentences, the program makes use of an anchor word list that contains word pairs of the languages in question. The one used here was not originally made for Portuguese and English, but was adapted from the English-Norwegian anchor word list. In the alignment process, a value is given to the combination of sentences based on matches in the word list. The program goes through the texts and reads chunks of sentences in each language, resulting in matrices in which the program seeks the highest values for a match between sentences. In addition to the anchor word list, these values depend on the number of characters within the s-units/sentences in each language, on special characters (such as ?!;, etc.), on proper names, and on cognates. One of the strengths of the program is that it does not require any preprocessing in the form of "hard regions", e.g. paragraph alignment, and therefore can get back on track after an alignment error (or in spite of a translation discrepancy).

To take an example, we could imagine the following chunks of text to be aligned:

English original (extract from Doris Lessing's *The Good Terrorist*)	Portuguese translation by Bernardette Pinto Leite
<s>She faced him, undefiant but confident, and said, "I wonder if they will accept us?"</s> <s>And, as she had known he would, he said, "It is a question of whether we will accept them."</s> <s>She had withstood the test on her, that bony pain, and he let her wrist go and went on to the door.</s>	<s>Ela encarou-o, sem desafio mas confiante, e perguntou:</s> <s>— Achas que nos aceitam?</s> <s>— E, conforme sabia que Jasper responderia, este retorquiu:</s> <s>— É tudo uma questão de nós os aceitarmos a eles.</s> <s>Alice resistira ao teste sobre a sua pessoa, à dor óssea, e ele largou-lhe o pulso e dirigiu-se para a porta.</s>

Figure 1. Texts to be aligned

We would expect that some words in the English extract would match some of the Portuguese words in the anchor word list; these matches would in turn enhance the possibility of the program linking the correct sentences. For the second English sentence in the extract above, for instance, we will get the following matches in the word list:[3]

and e
is, 's / é, está
question* / pergunt*, quest*
we nós
accept* aceit*
them / lhes, os, as

The TCA assigns a unique identification to each s-unit with its corresponding s-unit(s) in the translation. When the original sentence has only one corresponding sentence in the translation, we get a 1:1 correspondence. When the original sentence has been translated into two sentences, we get a 1:2 correspondence:

```
<s id=DL2.1.s16 corresp='DL2TP.1.s17 DL2TP.1.s18'>And, as she had known he
would, he said, "It is a question of whether we will accept them."</s>

<s id=DL2TP.1.s17 corresp=DL2.1.s16>&mdash; E, conforme sabia que Jasper
responderia, este retorquiu:</s>
```

```
<s id=DL2TP.1.s18 corresp=DL2.1.s16>&mdash; É tudo uma questão de nós os
aceitarmos a eles.</s>
```

Each text has a unique code, normally starting with the authors' initials, e.g. a text by Doris Lessing has a code DLx. All texts will be referred to by their code.

3. Human intervention required

The program handles 1:1, 1:2, and 1:0 correspondences, i.e. one s-unit matching one, two or zero s-units in the translation; the latter two have to be checked manually. The remaining matrices, containing 1:1 correspondences only, are assumed to be correct and are not systematically checked. Hence, the file that is proofread only includes matrices that do not contain 1:1 correspondences throughout.

Table 1 shows the percentage of matrices that the proofreader has to check, an average of 48.8%, for the 16 English-Portuguese texts. This apparently discouraging percentage needs some explanation since it does not reflect the actual manual intervention that takes place. The picture becomes skewed simply because we immediately associate half of the matrices with half of the sentences in a text. This is not the case, however. Each matrix contains approximately 10-12 sentence pairs, and as will be shown in the matrix below (Figure 2), the proofreader will only have to investigate the s-units that are not 1:1 correspondences. This is to say that although the proofreader has to investigate almost half of the matrices, he will not have to investigate half of the text/s-units, but merely a small percentage.

Table 1. Number of matrices to check

Text	Matrices in output file[4]	Matrices to check	%
ABR1	120	78	65
AH1	131	98	74
AT1	113	30	27
BC1	94	39	41
DL1	90	54	60
DL2	139	100	72
FF1	88	76	86
JB1P	97	32	33
JB1PP	97	32	33
JH1	61	20	33
MA1	74	6	8
NG1	74	37	50
PDJ3	108	49	45
RDO1	143	52	36
ST1	128	87	68
WB1	87	12	14
Total	1,643	802	48.8

To take an example, three of the sentence pairs in the matrix in Figure 2 have to be manually checked, and if necessary, corrected. This can be seen from the low values given in the diagonal of the matrix below (top left to bottom right). Moreover, the numbers below the matrix reflect this; the English sentence 4, for instance, is said to correspond to 0, sentence 5 to Portuguese sentences 4+5, etc. It can be seen, then, that

sentence 4 in the English original is not linked to any sentence in the translation, and wrongly so. We will therefore have to match it to the translation – sentence 3 in Portuguese – manually, correcting the alignment to a 1:2 correspondence. The other two 1:2 correspondences found in the matrix did not have to be corrected.

```
                84   20   71   51   23   56   48  106  133   51
                  1    2    3    4    5    6    7    8    9   10  Port. trans.
               ---------------------------------------------------
  1   97 I      6    0    1    2    0    0    0    2    1    1
  2   18 I      1    2    0    0    0    0    0    1    0    0
  3   58 I      2    0    0    1    0    1    0    2    2    0
  4    7 I      0    0    0    0    0    0    0    0    0    0
  5   78 I      2    0    1    4    3    1    2    3    1    0
  6   85 I      3    1    1    4    2    2    6    3    1    3
  7   96 I      2    2    1    3    1    2    1    7    2    1
  8  124 I      1    0    0    1    0    1    0    4    6    0
  9   51 I      1    0    1    2    1    2    2    1    1    2
 10  164 I      1    1    1    4    2    2    2    6    2    0
 .
Eng. orig.
           1,1 2,2 3,3 4,0 5,4+5 6,6+7 7,8 8,9 9,10
-------------------------------------------------------------------------------
 1: <s>For her part she did not have to be told that she was wearing <hi
    rend=italic>her look</hi>, described by him as silly.</s>  (DL2.1.11)

 1: <s>Quanto a ela, sabia que estava com o <hi rend=italic>seu olhar</hi>,
    que ele descrevia como de aparvalhado.</s></p>  (DL2TP.1.12)
-------------------------------------------------------------------------------
 2: <s>"Stop it," he ordered.</s>  (DL2.1.12)

 2: <p><s>&mdash; Pára &mdash; ordenou ele.</s>  (DL2TP.1.13)
-------------------------------------------------------------------------------
 3: <s>His hand shot out, and her wrist was encircled by hard bone.</s>
(DL2.1.13)

 3: <s>Estendendo a mão, apertou com força o pulso da rapariga, causando-lhe
dor.</s></p>  (DL2TP.1.14)
-------------------------------------------------------------------------------
 4: <s>It hurt.</s>  (DL2.1.14)


-------------------------------------------------------------------------------
 5: <s>She faced him, undefiant but confident, and said, "I wonder if they
will accept us?"</s>  (DL2.1.15)

 4: <p><s>Ela encarou-o, sem desafio mas confiante, e perguntou:</s></p>
(DL2TP.1.15)
 5: <p><s>&mdash; Achas que nos aceitam?</s></p>  (DL2TP.1.16)
-------------------------------------------------------------------------------
 6: <s>And, as she had known he would, he said, "It is a question of whether
we will accept them."</s></p>  (DL2.1.16)

 6: <p><s>&mdash; E, conforme sabia que Jasper responderia, este
retorquiu:</s></p>  (DL2TP.1.17)
 7: <p><s>&mdash; É tudo uma questão de nós os aceitarmos a eles.</s></p>
(DL2TP.1.18)
-------------------------------------------------------------------------------
 7: <p><s>She had withstood the test on her, that bony pain, and he let her
wrist go and went on to the door.</s>  (DL2.1.17)

 8: <p><s>Alice resistira ao teste sobre a sua pessoa, à dor óssea, e ele
largou-lhe o pulso e dirigiu-se para a porta.</s>  (DL2TP.1.19)
-------------------------------------------------------------------------------
 8: <s>It was a front door, solid and sure of itself, in a little side
street full of suburban gardens and similar comfortable houses.</s>
(DL2.1.18)
```

```
 9: <s>Era uma porta de entrada sólida, segura, situada numa ruazinha
secundária com jardins de subúrbios e casas semelhantemente
confortáveis.</s>  (DL2TP.1.20)
-------------------------------------------------------------------------------
 9: <s>They did not have slates missing and broken windows.</s></p>
(DL2.1.19)

10: <s>Não lhes faltavam telhas nem tinham vidros partidos.</s></p>
(DL2TP.1.21)
-------------------------------------------------------------------------------
```

Figure 2. Example of a matrix calculated by the Translation Corpus Aligner[5]

After the percentage of matrices to be checked had been calculated, the next step was to find out how many corrections one actually had to make. Table 2 gives the percentages of corrections that were made after alignment.

Table 2 presents, for each text, the number of s-units that need to be corrected after alignment with the anchor word list. Since not all s-units have actually been inspected by the proofreader (the percentage of matrices was shown in Table 1), the two rightmost columns give the number of s-units inspected manually, and the corresponding correction percentage.

Table 2. Number of corrected s-units after running the program with the anchor list

Text	Corrections	Total number of s-units	Corrected s-units (% of total)	Number of s-units in the matrices inspected[6]	Corrected s-units (% of inspected)
ABR1	33	1,139	2.9	740	4.4
AH1	42	1,263	3.3	934	4.5
ATI	19	1,102	1.7	297	6.4
BC1	25	893	2.8	366	6.8
DL1	11	855	1.3	513	2.1
DL2	61	1,307	4.7	941	6.5
FF1	48	713	6.7	613	7.8
JB1P	12	934	1.3	308	3.9
JB1PP	23	927	2.5	305	7.5
JH1	15	584	2.6	192	7.8
MA1	2	730	0.3	58	3.4
NG1	12	702	1.7	351	3.4
PDJ3	14	1,041	1.3	468	3.0
RDO1	42	1,396	3.0	502	8.4
ST1	50	1,204	4.2	818	6.1
WB1	5	727	0.7	101	5.0
Average			2.9		5.5

We see that the proofreader has to make changes in about 5.5% of the s-units inspected, which corresponds to about 2.9% of all s-units present in the corpus. Again it should be pointed out that the number of s-units found in the matrices that are inspected is considerably lower than the number of s-units actually inspected.

4. The importance of the anchor word list

We now proceed to evaluate the importance of the anchor word list, by comparing the amount of revision and modification required when running the alignment program with and without language specific information (that is, with an empty anchor list). In order

to estimate the differences, we ran the program with and without the anchor word list and then compared automatically the differences with the final (post-edited) version.

Table 3. Number of differences in the alignment of English-Portuguese texts

Text	No. of "skips"[7]	No. of differences (final version vs. raw version with anchor word list)	No. of differences (final version vs. raw version w/o anchor word list)	No. of differences (with vs. w/o anchor word list)
ABR1	8	33	139	126
AH1	3	44	111	88
AT1	4	27	38	39
BC1*	2	38		
DL1	2	14	31	30
DL2*	6	64		
FF1*	22	60		
JB1P*	3	18		
JB1PP	5	31	45	50
JH1	2	25	19	35
MA1	0	4	12	14
NG1*	0	15		
PDJ3	2	19	11	26
RDO1	2	67	72	65
ST1*	3	68		
WB1	8	6	14	9

The results are shown in Table 3. The texts marked with a star (six out of sixteen) could not be aligned without the anchor word list. Further, Table 3 shows the differences between the final, proofread versions of the ENPC texts and the versions with and without the anchor word list prior to revision.[8]

Due to the simplicity of the (programming) approach (only comparing the target part), missing s-units in the Portuguese translations, such as the (rare) example in Figure 3, were not identified:

```
6: <s>If we got married, we could no longer go back, whether we wanted to or
not.</s>  (ABR1.1.1.242)

 6: <s>Se nos casássemos, não poderíamos mais voltar, quer quiséssemos ou
não.</s>  (ABR1TP.1.247)
--------------------------------------------------------------------------
 7: <s>Neither he nor I.</s>  (ABR1.1.1.243)

--------------------------------------------------------------------------
 8: <s>He white; I coloured.</s>  (ABR1.1.1.244)

 7: <s>Ele era branco, eu mestiça.</s>  (ABR1TP.1.248)
--------------------------------------------------------------------------
```

Figure 3. Example of missing Portuguese translation, overlooked by the `compara_alinhamento.pl` program

It is clear that the English-Portuguese anchor word list does help the alignment program and reduces the number of changes to be made by the proofreader, though perhaps not as markedly as one might expect. However, the fact that 6 out of the 16 texts did not make it through alignment indicates that the program to a large extent depends on the anchor word list, and not only on the other factors mentioned in Section 2.

By using the anchor word list, the percentage of sentences to be corrected drops from 4 to 2% of all the sentences that are manually inspected (which, in turn, can be considered approximately one eighth of all sentences).[9] Again, let us stress that the anchor word list was not originally made for the language pair English-Portuguese. Neither of the anchor word lists was corpus driven; the original anchor list, for English-Norwegian, was manually encoded based on the intuition of the linguists working on the ENPC previous to the choice of the actual texts.[10]

5. Considering the anchor word list in detail

Would more attention to the English-Portuguese pair pay off? What would one do in order to optimize the performance for this language pair, and what would be the net gain? This is what we set out to test in the next step.

We have thus created a program which, for each English source text and its corresponding Portuguese translation, counts

- the number and percentage of the English anchor entries in the English text
  - compared to the entries in the anchor list
  - in terms of the s-units the matches refer to
- the number and percentage of the Portuguese anchor entries in the Portuguese text
  - compared to the entries in the anchor list
  - compared to the total number of target s-units
- the actual successful matches (i.e., the cases where both members of the anchor pair occur in a translation pair)
- the ratio of successful matches vs. all possible matches
  - in terms of the occurrences of the source member of the anchor list
  - in terms of the occurrences of the target member of the anchor list

In order to compute these numbers, several decisions have to be made:

First of all, the anchor word list is unwrapped from the source side, i.e., the cases of A,B / C are transformed into A / C and B / C, resulting in 1,022 pairs (from an original anchor list containing 882 lines). On the other hand, this is not done for the Portuguese side, since it is understood that E / F, G would succeed in either case. In Figure 4, we provide some illustration of what the unwrapping does:[11]

is, 's / é, está	's / ((é)|(está)) is / ((é)|(está))
English* / ingl*	English.* / ingl.*
became, becom* / torn*, volt*, fic*	became / ((torn.*)|(volt.*)|(fic.*)) becom.* / ((torn.*)|(volt.*)|(fic.*))
has, have, 've / tenho, tens, tem, temos, têm	has / ((tenho)|(tens)|(tem)|(temos)|(têm)) have / ((tenho)|(tens)|(tem)|(temos)|(têm)) 've / ((tenho)|(tens)|(tem)|(temos)|(têm))[12]
7*, seven / 7*, sete	7.* : ((7.*)|(sete)) seven : ((7.*)|(sete))

Figure 4. Original and modified anchor list

Then, pattern matching is done so that the matches are only counted in word contexts. Parts of words were not counted as successful matches (so *are* does not succeed in *mare*, for example), except when the pattern expression explicitly says so (like in *becom**).

In addition, the matching ignores case. This is justified due to the low probability of occurrence of non-capitalized instances of words that require capitalization (such as **english* or **inglaterra*).

The most important decision, as well as the one which may be less obvious, is that we only count 1:1 matches as being successful. That is, "successful pairs" in Table 4 below include only those in which the source member is found in a source s-unit which has **one** single sentence corresponding to it and in which, moreover, (one of) the target member(s) of the anchor word list is found. The reason for this restrictive computation of anchor list successes is twofold: If mappings of 1:2 or 1:3 were found,

- they would have been subjected to the revisor's consideration anyway
- it would be considerably more difficult to quantify both the success (should one use a pondered average of target s-units instead of natural numbers?) and the usefulness of the information in the anchor word list[13]

Finally, it should be noted that the anchor list included the entry *&mdash / &mdash*, which might not look specific to the English-Portuguese pair. However, some punctuation – and its translation – is language specific (Santos, 1998b), and therefore the anchor pair including *&mdash* was kept in the list for evaluation.[14]

5.1 Looking at the occurrences of anchor items in the corpus

Table 4 presents the quantitative results for each pair of texts. As far as the English matches are concerned, the first column displays how many members of the anchor list are actually present in the source text (the percentage relative to the total number of anchor pairs is given in the second column). The number of occurrences in the text of the source expressions of the anchor word list is given in the third column under "English matches". The number of different (target) s-units corresponding to some match of the English expressions in the source text is shown in the fourth column (i.e., more than one match of the same English expression per target s-unit is disregarded). As far as Portuguese matches are concerned, the corresponding information is provided: The next two columns give the number and percentage of target expressions of the anchor list which were found in the target texts, together with the number of actual matches. The most interesting information is found under the heading "Successful pairs", which displays how many times pairs in the anchor list were actually found in a translation pair, and what the percentage is relative to the simple occurrence of one element of the pair.

The success matches correspond roughly to a fourth of the source units in which they are found and to less than a tenth of the target units in which they appear. I.e., the relevance of the target expressions seems to be considerably lower than that of the source expressions. This can be explained by several distinct factors. First of all, the English terms were originally chosen with a view to finding a good translation correspondence with respect to Norwegian. This is not necessarily the case for the English-Portuguese pair. Then, the linguist adapting the list[15] failed to note that some target expressions were too general (matching, for example, unrelated and very frequent Portuguese prepositions – example in (4) below), which obviously diminishes the percentage of relevant pairs. Note that this does not necessarily diminish the program's performance, since there is no reason to suppose that the TCA looks for every occurrence in the target language, as we did here for evaluation purposes.

Table 4. Coverage of the anchor list for the English-Portuguese corpus

Text	English matches				Portuguese matches			Successful pairs		
	anchor list		s-units		anchor list		s-units	number	source	target
ABR1	695	68%	7,194	6,454	780	76%	50,226	3,585	49.8%	7.1%
AH1	655	64%	6,865	6,206	723	71%	51,147	2,981	43.4%	5.8%
AT1	624	61%	5,988	5,361	739	72%	48,585	3,177	53.1%	6.5%
BC1	700	68%	4,646	3,933	795	78%	38,381	2,500	53.8%	6.5%
DL1	609	60%	6,400	5,783	716	70%	45,860	3,328	52.0%	7.3%
DL2	623	61%	7,014	6,375	739	72%	48,279	2,761	39.4%	5.7%
FF1	690	68%	6,063	5,246	776	76%	39,095	2,505	41.3%	6.4%
JB1P	616	60%	6,056	5,434	737	72%	47,353	3,507	57.9%	7.4%
JB1PP	616	60%	6,063	5,437	758	74%	42,989	2,866	47.3%	6.7%
JH1	623	61%	5,040	4,417	743	73%	39,991	2,996	59.4%	7.5%
MA1	615	60%	5,448	4,833	735	72%	39,231	3,011	55.2%	7.7%
NG1	675	66%	6,536	5,857	746	73%	43,566	3,717	56.9%	8.5%
PDJ3	718	70%	7,632	6,910	803	79%	55,941	4,072	53.4%	7.3%
RDO1	538	53%	6,522	5,980	665	65%	43,813	3,880	59.5%	8.8%
ST1	692	68%	6,024	5,318	786	77%	49,826	2,781	46.1%	5.6%
WB1	629	62%	4,662	4,030	728	71%	32,463	2,675	57.4%	8.2%
Average	645	63%			748	73%			51.6%	6.6%

In any case, we find a significant number of matches, even if several pairs (at least around a fourth) do not appear in the files. If we divide the number of successes by the number of pairs whose source expression is present in the source text, we get on average 5 matches per pair. Later on we will look more closely at the significance of some elements of the anchor list.

Still, the number of successful pairs in Table 4 does not really offer a good illustration of the contribution of the anchor list, since they take each success independently of the place it occurs in. For example, for a particular s-unit there might be 10 successful matches with ten different pairs, while for another not a single one. This means that the values in the last two columns do not express the percentages of s-units that received positive match in the anchor list, but simply the percentage of occurrence of (source or target) expressions that have been useful for alignment.

So, we made our program also count the (target) s-units for which there was one or more matches for any anchor pair (i.e., the s-units which corresponded to success in finding both the source expression and the target expression). The program also provided the percentage of s-units in terms of the number of translation pairs in the file. These results, which are much easier to interpret, are displayed in Table 5: The first column shows the number of matching anchor-list pairs (after restructuring), and the percentage is given in column 2. The third column shows the number of s-units which had one or more successful matches in the anchor list, and column 4 lists the total number of translation pairs in each file (counted in the target file) for ease of reference. Column 5 is simply the result of dividing the value of column 3 by that of column 4, giving the percentage of translation pairs with successful matches in the anchor list.

We see from this table that the great majority of sentences have hits in the anchor pairs, which may indicate that, when a pair actually occurs in the text, it gives some positive result for the alignment process.

Table 5. Coverage of the successful anchor pairs in terms of anchor list and s-units

Text	Successes				
	anchor list		s-units	total	%
ABR1	563	55%	1014	1139	89%
AH1	488	48%	1069	1263	85%
AT1	496	49%	1030	1102	93%
BC1	557	55%	823	893	92%
DL1	492	48%	783	855	92%
DL2	482	47%	1120	1307	86%
FF1	512	50%	706	713	99%
JB1P	479	47%	882	934	94%
JB1PP	464	45%	879	927	95%
JH1	490	48%	572	584	98%
MA1	508	50%	711	730	97%
NG1	526	51%	653	702	93%
PDJ3	586	57%	970	1041	93%
RDO1	430	42%	1221	1396	87%
ST1	521	77%	870	1204	72%
WB1	515	50%	719	853	84%
Average	506	50%			91%

5.2 Looking at some anchor pairs in particular

We expect, however, that not all anchor pairs have the same import, and in fact we see from Table 5 that, on average, only half of them work for each text. That is why we wanted to take a closer look at the results.

From a rough inspection of the results, and along with more accurate pairs, the two following cases were easy to detect:

- systematic translations were forgotten (like the weapon sense of *arm** or the *artist** continuation of *art**)
- translation involves too much restructuring, due to language differences, so that lexical clues are not the right place to look (as is the case of *I* or *be*).

In order to look more closely at the results, we selected several frequent cases – pairs that were most successful (in absolute terms) – and looked at their distribution in the different texts. Table 6 shows the number of occurrences in the English source and those that corresponded to a match in the Portuguese translation. Texts in the Brazilian variant are identified in bold in the table.

Table 6 shows that some pairs are more useful than others, and also how this may vary for individual texts. Although we have to leave a detailed discussion of these results for another forum, we provide here some preliminary comments.

One case that is worth discussing is *I/eu*, which is interesting in that *eu* is considerably more rare in Portuguese (being a null subject language) than *I* in English. One can, however, notice that there is a considerable difference in utility (or translation correspondence) from text to text, and even note that, as expected, in the Brazilian variant there is a less marked tendency to dispose of the personal pronoun than in European Portuguese.

The opposite happens with dashes (coded *—*) in terms of relative frequencies: dashes are used much more frequently in Portuguese than in English. It is

interesting to argue for a different behaviour of the aligner in such cases, i.e., a different weighting of pairs related to criteria such as these. (The lower the frequency of the target item, the higher the probability that, being there, it is a translation of the source term.)

Table 6. The distribution of some frequent anchor pairs

pair	AB	AH	AT	BC	D1	D2	FF	JB	JB'	JH	MA	NG	PD	RD	ST	WB
&mdash	57	65	53	66	66	36	8	37	37	8	7	130	17	436	20	36
&mdash	247	454	343	305	236	507	12	47	130	36	33	127	210	445	389	234
	16	25	46	50	22	22	10	30	33	7	6	107	7	426	4	32
's	82	75	38	2	40	71	11	98	92	0	69	50	67	62	39	16
é, está	504	520	300	304	320	432	100	576	608	236	584	264	504	316	504	144
	30	35	12	2	21	27	3	55	57	0	33	29	34	33	15	11
I	238	171	73	84	43	115	39	316	316	11	217	110	124	322	80	205
eu	105	61	32	20	11	39	1	85	122	4	37	53	35	109	43	71
	87	40	19	17	8	19	1	62	104	3	32	43	22	94	30	63
be	65	92	41	21	65	66	64	54	54	48	38	61	80	24	61	19
ser, estar	172	116	132	60	152	120	84	172	248	160	116	172	172	68	220	60
	18	17	14	5	13	9	9	19	29	15	11	24	20	3	8	11
been	62	45	33	21	53	42	62	28	28	35	13	41	55	19	45	24
sido,	64	36	40	64	72	48	28	40	64	72	32	36	100	68	84	12
estado	13	0	7	7	10	4	5	4	11	10	4	5	18	6	7	2
could	29	19	48	18	33	41	45	25	25	54	13	33	68	46	32	21
podia.*, etc.	132	120	216	96	114	66	168	126	138	180	42	222	180	150	108	90
	12	10	28	8	7	6	18	10	9	24	6	20	25	19	9	10
is	65	51	14	41	29	47	25	60	60	53	130	25	45	19	35	21
é, está	504	520	300	304	320	432	100	576	608	236	584	264	504	316	504	144
	34	29	12	26	13	17	7	38	41	28	69	17	24	15	16	11
his	81	81	167	142	66	57	157	59	59	158	37	95	153	95	99	94
seu*,	220	272	488	364	264	212	268	124	364	360	136	488	496	272	208	224
dele*	19	12	64	52	24	13	37	7	23	61	12	44	60	25	15	26
there	52	54	42	23	47	44	43	38	38	47	64	58	51	79	23	31
lá, ali	144	88	84	32	124	120	28	172	68	36	80	112	140	180	152	68
	14	11	5	2	14	7	2	11	10	8	8	21	10	18	3	6

It is, however, also possible to see great variation without having either a significant frequency reduction or increase of the corresponding terms in the two languages, as the cases of *been* and *sido/estado* show. More detailed studies of the subtleties of the translation into Portuguese of *could* and of *there* can be found, respectively, in Santos (1998a) and Ebeling & Oksefjell (1998).

5.3 Looking at some instances of successful matches

One thing worth noticing is that, even though much of the data presented above could be explained with some linguistic ingenuity, it is not the case that all instances that were counted as successful matches are actually linguistically motivated. In fact, it was easy to detect three interesting situations from a crude inspection of the results:

First, even though the result might be correct, some of the resulting pairs "succeeded" without any sort of linguistic motivation; cf. (1) - (4):

(1) little / pequen.*?: *À sua frente, pelo lado esquerdo do caminho, ia um grupo de crianças, a mais velha a empurrar um carrinho, com duas crianças mais pequenas, uma de cada lado, agarradas ao varão.* (Translation of: Ahead of him, trudging along on the left of the path, was a little group of children, the eldest girl wheeling a pushchair with two smaller children, one each side of her, clutching the bars, PDJ3)

(2) be / ((ser)|(estar)): *Lá um* ***ser*** ('being', n.) *humano sempre será um estranho, jamais "pertencerá".* (Translation of: In there, one will always **be** a stranger, will never "belong", ABR1)

(3) look.*? / olh.*?: *Não há quebra-luz, apenas a lâmpada por cima que dá à minha cara um aspecto pálido e doentio, com* ***olheiras***.(Translation of: There 's no shade on the light, just a bare bulb overhead, which makes my face **look** pallid and ill, with circles under the eyes, MA1.)

(4) couple.*? / par.*?: *Depois de alguns anos* ***para*** ('for') *se estabelecer, ele começa a escrever.* (Translation of: After a **couple** of years to settle down he begins to write again, ABR1).

Secondly, it was also noted that the use of the Kleene star, instead of a more rigid morphologically closed list, allowed for correct matches which would certainly not have been listed by a linguist (due to considerations of frequency), as is the case of (5)-(6). Incidentally, a particularly valuable feature of the anchor list format was the ease of specifying cross-categorial translations (i.e., part-of-speech change in the translation).

(5) buil.*? / constru.*?: *Terminada a obra, os* ***construtores*** *serão mortos.* (Translation of: Once the job is finished the **builders** are killed, ABR1.)

(6) crim.*? / crim.*?: *Então, perto do final de 1347, uma pequena frota de cerca de uma dúzia de galeras genovesas chegou à Sicília vinda de um lugar distante, talvez da* ***Criméia****, e em poucos dias a população de Messina começou a morrer às centenas.* (Translation of: Then, towards the end of 1347, a small fleet of about a dozen Genoan galleys arrived in Sicily from somewhere far away, perhaps the **Crimea**, and within a few days the people of Messina began to die in their hundreds, ABR1)

Thirdly, example (7) displays a match which works despite the linguistic data it is supposed to mirror and/or despite the inaccurate translation. In fact, *earlier* in (7) was translated simply by *cedo* ('early'), leaving out the comparison (a literal translation would use *mais cedo*), but the fact that one has chosen to simply list the absolute form made the anchor pair more useful.

(7) earli.*? / cedo: *Ela trabalhava mais que qualquer um, acordava* ***cedo*** ('early'), *era a que ia dormir mais tarde.* (Translation of: She worked harder than anybody else, got up **earlier**, came to bed long after the others, ABR1)

In conclusion, it was possible to detect several pairs that led to "successes" without having been thought of as pairs beforehand. In some cases this increases performance, in others it diminishes it. Without looking at every pair that counted as a success, it is difficult to estimate the actual percentage of these cases.

The overall performance of the aligner did not appear significantly diminished, though, since, as mentioned above, there can be many possible unrelated "successes" for each s-unit. This rather illustrates that the TCA's approach, while not 100% linguistically motivated, seems to compensate on accounts of both simplicity and robustness.

6. Concluding remarks

Even though it was already known from the beginning that the TCA had been of extreme help in aligning the English-Portuguese texts of the ENPC, we think that the evaluation performed was interesting.

On the one hand, it measures the actual consequences of using this particular program for a given language pair, which is something that had not been done before. It is then up to the individual user to decide whether s/he should use this tool in his or her project.

On the other hand, this investigation also gives origin to some reflections concerning the evaluation process proper and the role of language dependency in NLP tools:

- One of the undeniable conclusions of the present study is, in fact, that it is a time-consuming and complex task to evaluate a particular program, and that it is often necessary to create auxiliary tools to evaluate it. It is, therefore, understandable that, in most cases, tool developers have no time to implement a thorough evaluation of the tools they develop because they employ their time in improving them (adding more functionalities, for example), not evaluating them. It is, therefore, understandable that, in most cases, tool developers have no time to implement a thorough evaluation of the tools they develop. Furthermore, it is often unclear whether the time used in this kind of evaluation is worthwhile, given that the program has a satisfactory behaviour for the task it was meant to perform.
- While adapting a particular language-dependent tool for other typologically similar languages, it seems that performance degradation is rarely so severe that it prohibits reuse of the tool. It is, however, seldom quantified what the changes are, and how well the "new" tool works when adapted to the new language (in this case, new pair of languages). That is why the present study may be interesting beyond the particular tool and pair of languages considered.
- Our preliminary conclusion is that even language-dependent tools rely to a lesser extent on languages than one would expect. Therefore, it is often feasible (and useful from an engineering point of view) to adapt that program to similar languages without considerable loss of performance. But see Santos (in press) for a more detailed discussion of language dependency and its methodological consequences.

As to the particular work reported here, we note that there are two other ways in which we could proceed for a thorough evaluation of the TCA for the English-Portuguese language pair:

- The first way would be to experiment with the constitution of the anchor list, creating new anchor lists, including for instance only the 20 most frequent pairs, or only the translations of the most frequent English content words, and compare the performances obtained.
- The second way would be to apply new texts and their translations to the TCA, and see whether this would result in any appreciable decrease in performance.

Unfortunately, both tasks will have to wait for another occasion, since they amount to considerable work.[16] Another revealing experiment would be to mimick the whole evaluation process for the English-Norwegian pair and compare the results.

7. Acknowledgements

We are grateful to Knut Hofland for making his program widely available and for answering our questions about it. In addition, the present paper was considerably improved by Stig Johansson's comments on both structure and content. We also thank Ana Frankenberg-Garcia for commenting on a previous draft.

Notes

[1] See http://www.portugues.mct.pt/.

[2] One of the English texts has been counted twice since two translations of it have been included in the corpus; one into European Portuguese and one into Brazilian Portuguese. For a complete list of authors and texts in the corpus see the ENPC homepage at http://www.hf.uio.no/iba/prosjekt/. For a more detailed description of the ENPC, see Johansson et al. (1999).

[3] This is the original format of the TCA wordlist. "*" means truncation, "/ "separates source and target items, "," separates items inside each language.

[4] In other words, the total number of matrices in the text.

[5] The first line of underscored numbers gives the number of characters in each Portuguese sentence, the second line the sentence number. The vertical italicized lines give the sentence number of each English sentence and its length in number of characters.

[6] We assume that the percentage of s-units inspected is equal to the percentage of matrices shown, therefore computing the present numbers by multiplying the total number of s-units by the percentage of matrices to check (displayed in Table 1). This is an approximation, because the same s-unit can belong to two contiguous matrices.

[7] It is important to note that in the process described so far, so called skip-attributes had already been inserted in the text files. Skip attributes are commands for the program to "skip" a particular sentence, and they are one of the few tags and attributes added by the ENPC project to the basic recomendations put forward by the Text Encoding Initative (Sperberg-McQueen & Burnard, 1994). The purpose of skip-attributes (in either language) is to mark that a sentence has not been translated, or has been invented. Although this is a sort of human intervention, it is hard to quantify, because it is usually inserted in an interactive mode when and if the program does not manage to continue. Here we have chosen to disregard them.

[8] These differences were computed automatically with the help of a simple Perl program (`compara_alinhamento.pl`), running after the Unix command `diff` was invoked between the resulting files.

[9] Assuming that for each matrix to be observed, 3-4 sentences have to be read by the human reviewer, this means a fourth of all sentences present in the matrices. Then, only half the matrices are revised: ½*1/4=1/8

[10] Knut Hofland (p.c.) revised and improved the list, using a program that computed a threshold of usefulness for very frequent items, and removing those which would not help the alignment.

[11] The "." and the parentheses in the reformulation are simply the rendition of the same semantics in Perl syntax and have no significance for the success of the pattern matching.

[12] From the unwrapping we see clearly that, given that Portuguese has more verbal inflections than English (whereas the opposite happens to Norwegian), one should have written the pairs `has/tem` and `have, 've/tenho, tens, temos, têm` instead.

[13] For example, one would expect the presence of the target member of the pair in a sentence contiguous to the right one (the one in the final alignment) as a measure of the negative influence a particular word-pair would have. When one sentence is aligned with two sentences, one of which would quantify as success in terms of our computations, one could expect a favourable import from the anchor word list if the alignment is right, and the opposite if it is not right. So, a good measure of the anchor list doings would have to take into account the percentage of e.g. 1:2 alignments that were correct. To do this would most probably require mimicking the way the program works, which is outside the scope of the present paper.

[14] In fact, even "invariants" such as numbers can actually result in a different translation. Floors are counted differently in different languages: ground floor in Portuguese is Norwegian first floor; soccer's first division in Norway is second division in Portugal; a person measured in feet will not have a

corresponding height in meters, etc. And Figure 4 reminds us that *seven hundred* does not get translated into *sete centos* but into *setecentos* (as opposed to Norwegian, which has two words as well).

[15] Incidentally, one of the authors of the present paper (D.S.).

[16] In order to compare the performance of the system with a revised version, one has to manually proofread the results first. For the case of the ENPC we had access to the proofread files, which is obviously not the general case.

References

Ebeling, J. & Oksefjell, S. 1998. On the translation of English there-sentences into Norwegian and Portuguese. What does a translation corpus tell us?. In Ydstie, J. T. & Wollebæk, A.C. (eds.), *Working Papers in Applied Linguistics 4/98*, Oslo: Department of linguistics, Faculty of Arts, University of Oslo. 188-206.

Hofland, K. 1996. A program for aligning English and Norwegian sentences. In Hockey, S, Ide, N. & Perissinotto, G. (eds.), *Research in Humanities Computing*. Oxford: Oxford University Press. 165-178.

Hofland, K. & Johansson, S. 1998. The Translation Corpus Aligner: A program for automatic alignment of parallel texts. In Johansson, S. & Oksefjell, S. (eds.), *Corpora and Cross-linguistic Research: Theory, Method, and Case Studies*. Amsterdam: Rodopi. 87-100.

Johansson, S, Ebeling, J. & Hofland, K. 1996. Coding and aligning the English-Norwegian Parallel Corpus. In Aijmer, K., Altenberg, B. & Johansson, M. (eds.). *Languages in contrast. Papers from a symposium on text-based cross-linguistic studies in Lund* (Lund, 4-5 March 1994). Lund: Lund University Press. 87-112.

Johansson, S., Ebeling, J. & Oksefjell, S. 1999. English-Norwegian Parallel Corpus: Manual. Oslo: Department of British and American Studies, University of Oslo, http://www.hf.uio.no/iba/prosjekt/ENPCmanual.html.

Santos, D. 1998a. Perception verbs in English and Portuguese. In Johansson, S. & Oksefjell, S. (eds.), *Corpora and Cross-linguistic Research: Theory, Method, and Case Studies*. Amsterdam: Rodopi. 319-342.

Santos, D. 1998b. Punctuation and multilinguality: Reflections from a language engineering perspective. In J. T. Ydstie & A.C. Wollebæk (eds.), *Working Papers in Applied Linguistics 4/98*, Oslo: Department of linguistics, Faculty of Arts, University of Oslo.138-160.

Santos, D. In press. Towards language-dependent applications, *Machine Translation* 14 (1999).

Sperberg-McQueen, M. C. M. & Burnard, L. (eds). 1994. *Guidelines for electronic text encoding and interchange. TEI P3*. Chicago & Oxford: Association for Computers and the Humanities / Association for Computational Linguistics / Association for Literary and Linguistic Computing.

Automatic proofreading for Norwegian: The challenges of lexical and grammatical variation

Koenraad de Smedt, University of Bergen <desmedt@uib.no>
& Victoria Rosén, University of Bergen <victoria@uib.no>

Abstract

In this paper we present some techniques, experiences and results from the SCARRIE project, which has aimed at developing improved proofreading tools for the Scandinavian languages. The focus is on methods which were used for spelling and grammar checking and particularly some novel analyses and treatments dealing with the extensive lexical and grammar variation in Norwegian Bokmål.

The major findings are that (1) since in Bokmål, lexical variants may differ with respect to grammatical features, stylistic replacement at the word level causes a need for grammar checking, and (2) the different systems for gender agreement in Bokmål can be handled in an economical way by a single grammar and lexicon if the features in the lexicon are interpreted dynamically depending on the subnorm or style preferred by the author.

1. Introduction

Among language technology applications, proofreading can be equally challenging as, for instance, machine translation. In a fair number of cases, errors in texts cannot be adequately corrected without understanding the intention of the author in the given context. In practice, however, automatic proofreading systems excel not by their understanding of the text but by their consistency and tirelessness in processing high volumes without becoming 'blind' to relatively simple errors as humans tend to become.

But even with limited expectations, the user may may find a proofreading system unacceptable if the number of false alarms is higher than the number of actual errors spotted, or if many suggestions for correction are inappropriate. It is therefore useful to invest in research aimed at improving the coverage of the system as well as the system's ability to propose corrections that are appropriate in the given context, whether grammatical or stylistic.

The SCARRIE project is a language technology project aimed at building high-quality proofreading tools for the Scandinavian languages (Danish, Swedish and Norwegian). The project was sponsored by the European Commission through the Telematics programme. The project ran from December 1996 through February 1999. The coordinator was WordFinder Software AB (Växjö, Sweden). The other main partners in the project were the HIT-programme at Universitetet i Bergen, Institutionen för lingvistik at Uppsala Universitet, Center for Sprogteknologi (København) and Svenska Dagbladet (Stockholm). Although the project aimed at eventual commercial exploitation, it did involve a great deal of linguistic and computational research.

At the end of the project, prototypes and evaluation reports were delivered for these languages. The prototypes correct simple misspellings and mistypings by means of advanced spelling and sound based matching criteria. They also have good coverage in their recognition of new compounds and derivations. Furthermore, they can detect repeated sequences, correct diacritical marks, correct words in the context of idioms and multi-word expressions, correct words based on different styles or norms, and perform limited grammar correction.

We will in the remainder of this paper only report on the Norwegian part of the project. Earlier publications (Rosén & De Smedt 1998, De Smedt & Rosén 1999) have highlighted different aspects of the linguistic and computational methodologies which are at the basis of SCARRIE for Norwegian. In this paper, we concentrate on the problems of proofreading for a language which shows rich variation not only in the lexicon but also in grammar. The specific problems related to grammar correction and style which are discussed below have to our knowledge never before been thoroughly researched with natural language processing methods.

2. Lexical and inflectional variants in Bokmål

Designing a system for automatic proofreading is difficult for any language, but Norwegian Bokmål presents a special challenge. Bokmål allows rich variation in the form of stems as well as inflectional endings. As we will see, this variation has grammatical consequences. First, we observe that many word stems in Bokmål have variants, as shown in for instance (1) and (2).

(1) *melk / mjølk* (milk)

(2) *gress / gras* (grass)

There is also variation in inflection, as exemplified in (3) and (4).

(3) *bok+en / bok+a* (book+DEF)

(4) *arbeid+et / arbeid+a / arbeid+de* (work+ed)

When computing the possible combinations of different stems and endings, we observe that the situation becomes more complex and the number of allowed variants increases, as demonstrated in (5).

(5) *melk+en / melk+a / mjølk+en / mjølk+a* (milk+DEF)

When compounding also enters the picture, word forms can easily have a dozen or more variants. At sentence level it is obvious that even more possible combinations may be found. Consider sentence (6) containing thirteen words; this sentence as a whole has no less than 165,888 possible spellings when all combinations of variants are enumerated.

(6) *De lavtlønte sykehjemsansatte ble helt utmattet og slukket tørsten med den surnete fløtemelken.*
(The low-paid hospital employees became totally exhausted and quenched their thirst with the soured cream milk.)

Not all combinations of variants are equally acceptable in all contexts, because variation is not free, but bound to more or less established subnorms within Bokmål. In other words, for almost all words that have variants, it is the case that the choice between them is not neutral, but depends on the author's style. Although the situation is vastly complex, we have in SCARRIE for Norwegian distinguished between three basic styles: radical, conservative and neutral. The stem *melk*, for instance, is conservative or neutral, whereas *mjølk* is radical; the ending *+en* is conservative or neutral, while *+a* is radical or neutral. Example (6) has only neutral variants; entirely conservative or radical variants of this sentence, as well as a great number of inconsistent combinations, can easily be constructed. As a final remark on basic styles, we mention that SCARRIE for Norwegian also handles a *school book norm (læreboknormalen)* in Bokmål, but this is another, quite complicated story which we will not go into here.

The fact that lexical items are associated with a norm or style value has a number of consequences. First, the user of a proofreading system should be able to state a preferred style. The system should be sensitive to that style so that whenever it makes a suggestion for a correction of a spelling error, it proposes a form that fits with the author's style. Second, we can observe that some forms are rarely or never used because they are infelicitous combinations of different styles, such as *mjølken* in (5), which combines a radical stem with a non-radical ending. Even though such forms may be allowed in Bokmål, they will need to be replaced under all major styles (conservative, neutral and radical) if consistency is to be achieved. Third, variants may have different grammatical features; this final complication is an important theme of this paper.

3. Lexicon

SCARRIE uses full-form lexicons which contain all inflectional forms of words except genitives (which are very regular). In order to restrict the system's suggestions for correction to those word forms that occur in the author's chosen style, it would be possible to construct separate lexicons for each subnorm. However, since there is considerable overlap between subnorms, this would be a wasteful and inflexible solution. Moreover, separate lexicons would not allow straightforward correction of word forms belonging to other styles than the author's stated preference. Therefore, one integrated lexicon was constructed with replacements depending on style. Table 1 presents a simple example, consisting of the lexical entries belonging to the lemma *bok* (book).

Table 1. Lemma for *bok* (without frequency information)

word form	*style code*	*compound codes*	*replacement*	*grammar code*
bok	N	N,sg,indef		N_f_sg_indef
boka	C2	N,sg	boken	N_f_sg_def
boken	C3	N,sg	boka	N_fm_sg_def
bøkene	N	N,pl		N_f_pl_def
bøker	N	N,pl,indef		N_f_pl_indef

The entries for the indefinite singular *bok* (book), plural definite *bøkene* (the books) and plural indefinite *bøker* (books) all have a style code N which means they are normal forms and do not need to be replaced under any styles. The entry for the singular definite *boka* (the book) specifies that under style code C2 (conservative), it should be replaced by *boken*. Conversely, the entry for *boken* specifies that under style code C3 (radical), it should be replaced by *boka*. In other subnorms, both word forms are acceptable and therefore never replaced. For forms with more variants, the coding in the lexicon can be quite complex; for more examples from the lexicon, we refer the reader to Rosén & De Smedt (forthcoming).

We focus now on grammar checking, which obviously relies on grammatical information associated with lexical entries. The last column in Table 1 contains grammar codes that are used by a parser which can for instance detect lack of agreement in the NP, as in (7).

(7) * *Den lille bøkene* (the little+SG+DEF books+PL+DEF)

Before discussing the grammar codes in the lexicon in more detail, the grammar correction mechanism itself will first be sketched.

4. Grammar correction in SCARRIE

Various approaches to grammar correction have been tried out for the various languages covered in the SCARRIE project. The system for Norwegian is based on the CORRie platform, which has a built-in LR parser based on augmented context free grammar (Vosse 1992, 1994). Grammar rules for Norwegian were written for use with this parser. The following kinds of grammatical errors can be automatically corrected by the Norwegian SCARRIE grammar:

1. Lack of gender, number and/or definiteness agreement between (a) determiner, adjective phrase and noun in NP, (b) subject or object and nominal or adjectival complement in S, and (c) noun and postposed possessive in NP.

2. Errors involving (a) the wrong sequence of verb forms in VPs and (b) finite vs. non-finite verb forms.

3. Errors involving case forms for object pronouns in topicalized position and for corresponding subject pronouns in inverted position.

Although native speakers of Norwegian would clearly recognize these kinds of errors, they are not uncommon as results of mistypings and editing routines and are occasionally overlooked by human proofreaders. An example of lack of gender agreement is (8), corrected as (9).

(8) * *Et morsomt gutt ler.* (A(neuter) funny(neuter) boy(masculine) laughs.)

(9) *En morsom gutt ler.*

Grammar correction of Norwegian in SCARRIE is based on the detection and correction of mismatches of grammatical features. *Error weights* attached to phrase structure rules make it possible not only to find such feature mismatches, but also to suggest corrections for them. Each feature on the right hand side of a phrase structure rule may have an error weight associated with it, the default being 1. A weight higher than 1 indicates that the feature 'carries more weight'. An example of such a rule is (10).

(10)
```
NP(Gender Number Definiteness NCase)
-> Det(Gender Number Definiteness:3 [dem quant])
   AP(Gender _ Number Definiteness)
   N(Gender:5 Number Definiteness NCase)
```

Trying to correct a feature mismatch by changing the gender of the noun will now produce a total weight of 5, whereas changing the gender of both the determiner and the noun gives a total of 2. The system chooses the analysis with the lowest error weight, and looks up the word forms *et* and *morsomt* in the lexicon. It will find other word forms in the same lemmas with the feature masculine, and can therefore suggest the correction in (9).

The features in the grammar rules refer to features associated with word forms in the lexicon. However, this coding in the lexicon (cf. the last column in table 1) is not straightforward. The reasons for this will become apparent after a discussion of systematic gender variation in Bokmål.

5. Gender systems

Besides the considerable variation in stems and endings, Bokmål has several systems for gender agreement. We can distinguish between three major gender systems. The most obvious lexical characteristic is that feminine singular nouns sometimes behave like masculine ones, both with respect to endings and agreement. This variation is schematically shown in table 2.

Table 2. Main gender systems in Bokmål

3 gender system	*2.5 gender system*	*2 gender system*
ei lita bok	*ei lita bok	*ei lita bok
*en liten bok	en liten bok	en liten bok
boka mi	boka mi	*boka mi
*boken min	boken min	boken min

The first two rows deal with the indefinite form. Here we see that the indefinite form *bok* occurs in all styles. However, it agrees with feminine determiners and adjectives in one system, while it agrees with masculine determiners and adjectives in the other systems.

The bottom two rows show the definite variants *boka* and *boken,* which are both acceptable in the 2.5-gender system. In the 2-gender system, *boka* is not acceptable, while *boken* is unacceptable in the 3-gender system. We have outlined above how lexical entries with replacements can deal with this variation depending on specified styles. In addition, however, we have to take care of agreement, just like we have to for the indefinite form.

The main question is, how can we achieve this variation of the treatment of gender, which not only seems to require different allowable word forms under different styles, but also different grammatical features for the same entry under different styles? One might think it was necessary to use multiple lexicons, multiple grammars, or both. We will show how in fact a more practical and economical solution was devised, consisting of a flexible interaction between a single lexicon and a single grammar.

This solution requires that lexical entries are coded appropriately to reflect the described variation. Unfortunately, the consequences of this variation were never taken care of by lexicographers before the need for a proper natural language processing treatment manifested itself. In Bokmålsordboken and in NorKompLeks, which the Norwegian SCARRIE lexicon is based on, all feminine words are coded as both *m* and *f.* Unfortunately, this does not differentiate between those nouns that are obligatorily *f* in a 3-gender system (e.g. *bok, jente*), and those that may be either *m* or *f* in such a system (e.g. *art, krokodille, nytte,* etc.). This coding does not allow for correct agreement in a 3-gender system.

6. Grammatical codes for gender

We wil now turn our attention to the way in which the codes in the lexicon (cf. the last column in table 1) are related to the features used in the grammar rules. We have opted to create new codes and add them to the SCARRIE lexicon of fully inflected word forms. Here we differentiate between the two classes mentioned before: only words like *krokodille* are treated as *m* or *f,* which means they have the full inflectional pattern of both genders. All other feminine nouns are treated as only *f* in the lexicon, except for

the form with masculine inflection (e.g. *boken*), which receives a special code *fm*, as shown in the last column of table 1.

However, the codes in the lexicon are not to be taken at face value; they are interpreted by subnorm-dependent *translation tables* that convert them to the feature structures required for grammatical analysis. For example, it could be specified that a code as in (11) is to be translated to the grammatical expression (12) which matches expressions in rules such as (10).

(11) `N_f_sg_indef`

(12) `N(f sg indef nocase)`

The effects of the different gender systems are achieved by using not just one translation table, but different translation tables dependent on the author's chosen style. An overview of the subnorm-dependent translations for the relevant entries of the lemma *bok* is shown in the table 3 (with the feature *nocase* omitted for simplicity).

Table 3. Style dependent translations of grammatical codes

word form	***code in lexicon***	***3 gender system***	***2.5 gender system***	***2 gender system***
bok	N_f_sg_indef	N(f sg indef)	N(m sg indef)	N(m sg indef)
boka	N_f_sg_def	N(f sg def)	N(f sg def)	N(m sg def) *
boken	N_fm_sg_def	N(m sg def) *	N(m sg def)	N(m sg def)

When we use the translation table for the 3 gender system, the code for *bok* in the lexicon gives rise to the value *f* for the gender feature. Using grammar rules like (2), this enforces agreement with a feminine determiner, as it should in this system. In a 2.5 or 2 gender system, the code gives rise to the value *m*. This enforces agreement with a masculine determiner.

Next, consider the entries for *boka* and *boken*. The forms marked with an asterisk are not acceptable in the given systems and will be replaced, as was discussed in an earlier section. The remaining forms are coded such that *boka* agrees with the feminine and *boken* with the masculine determiner.

7. Interaction between agreement checking and replacement

The two mechanisms described above, style dependent replacement in the lexicon and style dependent agreement checking in the grammar, each deal with specific aspects of the described variation. Still, it is not sufficient to specify these mechanisms separately. Rather, these mechanisms must interact in order to correct entire phrases such that not only the resulting word forms are allowed under the given subnorm, but also appropriate agreement results.

Consider the correction of the phrase *boken min* in radical Bokmål, for instance. The phrase is grammatically correct, but the inappropriate use of the word form *boken*

triggers correction. However, simply substituting *boka* for *boken* would result in an agreement error where there there previously was none: **boka min.*

Therefore, *after* a word form has been substituted, the sentence must be checked grammatically. Since substituting one word form for another may result in changes in grammatical features, the *new* features are used in the syntactic analysis. In the example given, this may cause detection, and subsequent correction of the lack of agreement. In this way, substitution of *boka* for *boken* triggers also the substitution of *mi* for *min,* resulting in the final correction to *boka mi.*

8. Parsing and grammatical correction

The usefulness of the approach taken will be shown with the help of a few examples of how sentence (13) is corrected in different styles.

(13) *Heimeleksen din er ferdig.* (Your homework is finished)

This example contains the word *heimeleksen,* which has a radical stem and a conservative ending. It will be corrected in different ways depending on style. A correction in style 2 (conservative Bokmål), as it appears in the output from SCARRIE, is given in (14).

(14) #1#Heimeleksen din er ferdig.
-- 1.Hjemmeleksen

In this correction, the radical form *heimeleksen* is replaced by *hjemmeleksen.* There is no grammatical error in this case. In style 3 (radical Bokmål), however, the same sentence is corrected differently. The word form *heimeleksen* must be replaced with *heimeleksa*, as shown in (15).

(15) #1#Heimeleksen #2#din er ferdig.
-- 1.Heimeleksa 2.di

This correction implies replacing a masculine form by a feminine form. Although the original sentence was grammatically fine, the replacement *heimeleksa* has a gender feature that now is in conflict with that of the determiner. Rules such as (10) detect such mismatches and the correction of *din* to *di* ensues.

A final parsing example (16) is meant to show how insufficient coverage in the grammar, together with massive lexical and structural ambiguity may lead to problems in grammar checking.

(16) *Resultatet er det vi har kalt for fiksering i problemløsning.*
(The result is what we have called fixation in problem solving)

Sentence (16), which is error free, nevertheless receives the suggestions for correction shown in (17).

(17) Resultatet er det vi #3#har #4#kalt for fiksering i problemløsning.
-- 3.har?4.kalte

Parsing this sentence results in no less than 28 trees, none of them error free. The

reading which the parser chooses for correction is one in which *kalt for* is analyzed as the NP *kalt fôr* (called lining). With a better coverage of the grammar, the parser should have chosen an error free analysis.

9. Results and discussion

The overall results of testing SCARRIE for Norwegian were very favorable compared to existing systems, as was reported in more detail in Rosén & De Smedt (forthcoming). Without giving further details on other test results, we mention that grammar checking was tested on a test suite containing 20 different NP agreement errors (of several types, including types discussed above), 12 VP errors and 32 style/subnorm errors. All except 2 style errors received perfect corrections.

However, the system's grammar checking exhibits considerable discrepancy between lab performance, which has shown great potential, and tests on realistic texts, which show poor reliability. The reasons why grammar checking performs poorly on authentic texts are the following:

1. The coverage of the grammar is too limited. The projected time for working on grammar checking was only 4 person months, while the actual time spent on it was less than 3 person months. Any project aiming at developing a truly wide coverage grammar from scratch should be measured in person years rather than months.

2. Lexical and syntactic ambiguity cause a large number of analyses of correct sentences. For sentences with errors, the number of possible analyses becomes even larger. It is very difficult for an automatic system to choose the 'proper' incorrect analysis for correction. We believe that this is a problem not only for our own approach, but for any grammar checking which is insensitive to meaning. We think it will also affect shallow parsing systems. Such systems will, if they are scanning for NPs, always run the risk of wrongly analyzing the kind of pseudo-phrase shown in example (16).

3. The grammar formalism used by the parser is limited, for instance in its treatment of long-distance dependencies. It is difficult to attain wide coverage without at the same time allowing unwanted rule interactions which result in spurious analyses.

10. Summary and conclusion

From a language technology perspective, we analyzed the problems that variation in Bokmål poses for proofreading and found new solutions that dealt with the problems in a systematic and linguistically motivated way. Some parts of the solutions implied adaptations of the underlying CORRie engine which was used for all languages involved in the project, while other parts were achieved by a creative and efficient design of the lexical and grammatical data for Norwegian.

In this paper, we concentrated on correction of NP agreement in Norwegian, for various reasons. First, an error corpus for Norwegian (Rosén & De Smedt 1998) revealed that a number of these errors indeed occurs in writing. Second, the CORRie parser which was

used has good feature-based mechanisms for handling agreement, which is at the core of our treatment of NPs. Finally, agreement is non-trivial in Bokmål due to the interesting variations and therefore its computational processing poses challenging research questions.

We have described two mechanisms which together handle the variation at the lexical and grammatical levels. One mechanism makes use of lexical replacement depending on style. The other mechanism is agreement checking using a robust LR parser and grammar. We have shown that in Bokmål, both mechanisms are necessary: lexical replacement in Bokmål is dependent on subsequent agreement checking, because variant word forms do not necessarily have the same grammatical features.

Of particular importance is the interaction of the grammatical and lexical levels for handling linguistic variation. By using translation tables dependent on style, we obtain a flexible interface between the lexicon and the grammar. In fact, multiple lexicons or multiple grammars are simulated in this way, which is a powerful feature.

Some remarks are to be made on the limitations of the system. First, grammar checking in SCARRIE for Norwegian slows the system down by a factor of ten compared to running a spelling check without using the parser. Second, even though the current grammar checking performs very well on construed examples, it is not reliable on authentic texts. Due to massive lexical and structural ambiguity, sometimes errors are not detected, or, even worse, they are corrected to something unintended. Therefore, realistic grammar checking is legitimately the subject of more in-depth research.

11. References

De Smedt, Koenraad & Rosén, Victoria 1999. Datamaskinell skrivestøtte. In: Birgitta Lindgren (ed.) *Språk i Norden 1999* (pp. 20-32). Oslo: Novus.

Landrø, Marit Ingebjørg & Wangensteen, Boye 1993. *Bokmålsordboka* (2nd ed.). Oslo: Universitetsforlaget.

Rosén, Victoria & De Smedt, Koenraad 1998. SCARRIE: Automatisk korrekturlesning for skandinaviske språk. In: Faarlund, J.T., Mæhlum, B. & Nordgård, T. (eds.) *Mons 7: Utvalde artiklar frå det 7. Møtet Om Norsk Språk i Trondheim 1997* (pp. 197-210). Oslo: Novus.

Rosén, Victoria & De Smedt, Koenraad, forthcoming. *Er korrekturlesningsevnen di god? Resultater fra SCARRIE. *Proceedings of MONS 8, Tromsø, Nov. 18-20, 1999.*

SCARRIE, Norwegian homepage: http://fasting.hf.uib.no/scarrie/

Vosse, Theo 1992. Detecting and correcting morpho-syntactic errors in real texts. In: *Proceedings of the Third Conference on Applied Natural Language Processing, Trento* (pp. 111-118). Association for Computational Linguistics.

Vosse, Theo 1994. *The word connection.* Enschede: Neslia Paniculata.

Word Alignment Step by Step

Jörg Tiedemann

Department of Linguistics, Uppsala University

joerg@stp.ling.uu.se

Abstract

In this paper the current stage of the Uppsala Word Aligner (UWA) is described. The system is developed within the project on parallel texts, PLUG, which has its focus on the analysis of bi-lingual text collections with Swedish either as the source or the target language. UWA comprises a set of knowledge-lite approaches[1] for word alignment and lexicon extraction. A distinctive feature is its modularity. In the article, the main principles of the alignment software are introduced, different configurations and approaches are described, and examples of alignment results are presented.

1. Introduction

Word alignment aims at the identification of translation equivalents between linguistic units below the sentence level within parallel text (Merkel 1999), mainly bilingual text (*bitext*). Those units include single-word units (*SWUs*) and multi-word units (*MWUs*) and will be referred to as *link units* further on. The basic terminology for describing parallel text and word alignment in this paper follows Ahrenberg et al (1999) and Ahrenberg et al (forthcoming). In particular, each word correspondence in the bitext describes a *link instance*, or simply a *link*. A pair of link units that is instantiated in the bitext will be referred to as *link type*. Word alignment systems usually assume segmented bitext (*sentence aligned bitext*). Common *bitext segments* are sentence fragments, sentences, and sequences of sentences that have corresponding units in the translation.

Depending on its purpose, a word alignment system attempts to maximize the number of discovered links (→ word instance alignment) (e.g. Ahrenberg et al 1998, Melamed 1999) or the number of extracted link types (→ bilingual lexicon extraction) (e.g Melamed 1995, Resnik and Melamed 1997, Tiedemann 1998a). Lexicon extraction aims at providing correct translations whereas word alignment has to deal with insertions, deletions, and other modifications within the bitext as well. Furthermore, word alignment systems may focus on specific types of link units, e.g. terms (Dagan and Church 1994, van der Eijk 1993) and collocations (Smadja et al 1996).

The task of word alignment is not trivial especially because it goes beyond simple one-to-one word correspondences in many cases. Multi-word units (MWUs) have to be handled due to the use of non-compositional compounds, associated idiomatic expressions, multi-word names and so on. The difference in compounding between different languages increases the difficulties with the identification of appropriate

correspondences further. In addition, the text type is decisive for the word alignment process. Technical text tends to include specific terms and simple structures that are translated directly while e.g. literary texts include many language-specific idioms.

Concurrently, Martin Kay's proposal for approaching machine translation can be applied to word alignment as well:

"The keynote will be modesty. At each stage, we will do only what we know we can do reliably. Little steps for little feet!" (M.Kay 1980)

The alignment of MWUs can be approached in different ways. Smadja et al (1996) propose the compilation of source language collocations using statistical co-occurrence measures (*static segmentation*). The appropriate correspondent is found by iterative extension of the link unit in the target language segment (*dynamic segmentation*). Another approach applies collocation lists for both languages, which have been compiled in advance from the bitext (Ahrenberg et al 1998, Tiedemann 1998). MWUs are then handled like single tokens for both languages. A third possibility is to expand link units iteratively for both languages in order to find the most appropriate link. Melamed (1997) uses iterative processing in order to optimize the underlying translation model. The iteration is alternated in order to cover MWUs in both languages.

The word alignment system, which is introduced in this paper, supports all the three approaches to text segmentation as far as contiguous phrases are concerned. The approach to dynamic segmentation differs from Melamed in the usage of ranked candidate lists instead of translation models. Furthermore, classified stop word lists are used for improving the result and reducing the search space.

2. The Uppsala Word Aligner (UWA)

The Uppsala Word Aligner is developed within the co-operative project on parallel texts, PLUG[2] (Sågvall Hein, forthcoming). The goal of PLUG is to develop, apply, and evaluate software for the alignment and generation of translation data. Word alignment is one of the major issues at hand. UWA is based on earlier studies on bilingual lexicon extraction (Tiedemann 1997, 1998a). It combines several knowledge-lite approaches to word alignment. The system is integrated into the Uplug toolbox (Tiedemann forthcoming), which provides a convenient environment for the work with modular corpus tools.

As mentioned earlier, word alignment walks with small feet. Therefore, the proposal is to combine different approaches, to collect available knowledge sources, and to reach the goal by little steps.

The principles of baby-steps

1. Prepare carefully before taking the first step.
2. Use all available tools that can help.
3. Check alternatives before taking the next step.
4. Take safe steps first; try risky steps later.
5. Remove everything that is in the way.
6. Improve walking by learning from previous steps.
7. Reach the goal with many steps rather than one big one.
8. Continue trying until you cannot get closer.

Based on these general principles, UWA was designed as a modular and iterative (rule 6+8) system. The bitext runs through initial pre-processing steps before the alignment starts (rule 1). Alignment candidates are collected from any appropriate source (rule 2). Candidates are ranked by their reliability, e.g. association scores (rule 3). The most reliable candidate is aligned first (rule 4). The alignment process is split into a sequence of separated steps (rule 7). Aligned link units are removed from the search space (5).

In the following the three main phases of the UWA are described: text segmentation, candidate collection, and alignment of link units.

2.1 Text Segmentation

UWA assumes sentence-aligned bitexts. However, an initial sentence alignment step can be added.

UWA provides modules for the work with static and dynamic text segmentation. In the case of dynamic segmentation, the text will be simply tokenized, i.e. segmented into SWUs and punctuation marks. In case of static segmentation, this phase accounts for a subsequence segmentation of the bitext into link units. It includes tokenization, the recognition of MWUs, and the actual segmentation of the text into link units. The recognition of MWUs can be automated. UWA applies iterative processing for the compilation of word collocations. The association between word units and their subsequent words is measured in terms of mutual information scores. As proposed in Ahrenberg et al (1998), classified lists of functional words are used to reduce search space and to exclude ungrammatical constructions. Consider the small example of classified stop words for English phrase generation, which is illustrated in figure 1.

Figure 1: Classified stop word lists (lower case).

```
skip token = '((,[,{,),],},\,',!,?)'
skip at = '(or,and,but,not)'
skip before = '(i,you,he,she,it,we,they)'
skip after = '(mine,yours,his,hers,its,ours,theirs)'
non-phrase-starter = '(my,your,his,her,our,their)'
non-phrase-ender = '(the,a,an)'
skip at string type = '(numeric)'
```

Stop words are divided into 6 types. '*Skip token*' items are not considered at all in any segmentation. Furthermore, the segmentation will stop at '*skip at*' tokens. They are considered to be single word units and the segmentation process continues with the subsequent token. '*Skip before*' defines link unit breaks in front of each instance of each word that is specified. Similarly, '*skip after*' defines breaks after each instance of words in the list. '*Non-phrase-starter*' and '*non-phrase-ender*' are not allowed in the beginning or at the end of any phrase, respectively. However, those words may appear within phrasal constructions as e.g. in 'in my mind' or 'in a row'. Note, that the definite article is allowed in the beginning of a phrase. In this way correspondences between definite forms of English and Swedish nouns can be recognized[3]. Furthermore, each category may include all tokens of a certain string type. In the example above, all numeric tokens will be added to the 'skip at' list. In cases of overlapping definitions the stronger restriction is chosen. In the current stage of the system only four of the classes above are used: 'skip token', 'skip at', 'non-starter', and 'non-ender'.

In figure 2 a sample of an automatically generated list of English collocations is presented. It is based on an English subtext from the PLUG corpus (Tiedemann 1998b) with about 66,000 words. The minimal frequency was set to 4.

Figure 2: Generated phrases with frequency>4 (case-folded).

MI	freq	collocation
10.039	4	the yom kippur
10.039	4	raymond aron
10.039	4	danny kaye
9.717	5	yom kippur
9.717	4	the golan heights
9.717	4	golan heights
9.454	6	world war ii
9.454	6	the mishkenot sha
9.454	4	lyndon johnson
9.454	4	american public opinion
9.395	4	justice cohn
9.231	7	tel aviv
9.231	7	mishkenot sha
9.231	5	the ottoman empire
9.231	5	ottoman empire

In the current stage, the system provides contiguous phrases only. Static text segmentation applies a simple left-to-right process. It starts with the left-most token in the bitext and looks for the longest valid link unit. The segmentation continues to the right of the last validated link unit until the complete bitext is processed. Here, single word units always represent valid link units and sentence breaks always mark the end of the current link unit.

2.2 Identification and Collection of Candidate Pairs

In this part the system compiles and collects translation equivalents. Sources are pre-compiled collections and generated lists of candidate pairs. In the current implementation, UWA applies the following sources:

- pairs of associated word units (applying co-occurrence measures)
- cognate lists (applying string similarity measures)
- single word bitext segments
- pairs of low frequency units
- machine readable bilingual dictionaries (MRBDs)
- previously aligned word pairs (iteration)

Collections of candidate pairs are compiled by investigations on the association between link units. UWA applies co-occurrence measures and string similarity scores in order to find alignment candidates. The number of possible candidates is reduced by some general restrictions in order to improve the performance:

link distance: link units have to occur within a certain distance between their positions in the bitext segment
string length: each link unit has to exceed a minimal length
length difference ratio (LDR): the ratio between the length of the shorter link unit and the length of the longer link unit has to pass a certain threshold value
string type: each link unit has to present a certain string type (e.g. 'contains at least one alphabetic character')
frequency: the number of occurrences of each link unit has to exceed a certain value (for co-occurrence measures only)
co-occurrence frequency: each pair of link units has to co-occur at least a certain number of times (for co-occurrence measures only)

The value of each of the parameters above can be adjusted to the type of investigation in progress. Certainly, string length and LDR should be restricted for investigations on string similarity, whereas frequency thresholds are important for co-occurrence measures.

UWA supports three *word association* scores: the Dice coefficient, mutual information, and t-score. The current investigations were focused on the application of the Dice coefficient.

$$Dice = \frac{2\,prob(S,T)}{prob(S) + prob(T)}$$

In our case S and T represent the link units in the source and the target language under consideration. The probabilities of S and T to occur in the text, and the probability of both units to co-occur in the same bitext segment (sentence alignment) can be estimated by appropriate frequency counts. Simple stemming functions are used in order to reduce the inflectional variety of words in different languages and to improve the statistical calculations.

String similarity can be measured by different metrics (Melamed 1995, Borin 1998). UWA uses the Longest Common Subsequence Ratio (LCSR). UWA applies dynamic programming for computing the length of the longest common subsequence (LCS) of two strings (Stephen 1992). This value, divided by the length of the longer string, provides a measure for string similarity between them. In figure 2, the LCSR calculation is illustrated. In the figure, the application of the algorithm with MWUs is demonstrated as well.

Figure 2: The longest common subsequence ratio of 'see example' and 'se exempel'.

	s	e	e		e	x	a	m	p	l	e
s	1	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2	2
	1	2	2	3	3	3	3	3	3	3	3
e	1	2	3	3	4	4	4	4	4	4	4
x	1	2	3	3	4	5	5	5	5	5	5
e	1	2	3	3	4	5	5	5	5	5	5
m	1	2	3	3	4	5	5	6	6	6	6
p	1	2	3	3	4	5	5	6	7	7	7
e	1	2	3	3	4	5	5	6	7	7	8
l	1	2	3	3	4	5	5	6	7	8	8

$$LCSR = \frac{length[LCS(S_1, S_2)]}{\max[length(S_1), length(S_2)]} = 8/11 \approx 0.72$$

Further investigations on string similarity metrics have been carried out (Tiedemann 1999) but they have not yet been applied in the word alignment process.

Another source of alignment candidates can be found in *single word bitext segments.* UWA considers each bitext segment with exactly one link unit in one language to be a valid alignment candidate.

Low frequency link units cannot be recognised by statistical association scores. However, they represent a large portion of general text corpora. UWA applies a simple heuristic in order to extract alignment candidates of low frequent text units. Assuming two frequency thresholds t_1 and t_2 with $t_2 < t_1$, the system removes all units that occur less than t_1 times in the complete text from each bitext segment. Now, each bitext segment with exactly one remaining link unit on each side is considered to be a valid alignment candidate, if both link units occur less than t_2 times. However, finding appropriate values for t_1 and t_2 is not trivial. Mainly, the distance between t_1 and t_2 is significant for the quality of extracted pairs.

Furthermore, *MRBDs* of any origin can be added to the collection of alignment candidates. Certainly, their quality is decisive for the quality of the word alignment. This includes that the chosen MRBDs should be suitable to the type of text under consideration.

As mentioned earlier, UWA supports dynamic text segmentation. If this alternative is chosen, the system generates all combinations of possible link units. The number of possible units can be reduced drastically by the use of classified stop word lists. The same principles as for static text segmentation are applied (except co-occurrence thresholds). Consequently, only contiguous phrases are identified. Association scores are computed for each possible link unit combination. This includes co-occurrence measures as well as string similarity measures. In this way, a list of alignment candidates is collected that includes all link unit combinations that pass a certain threshold value.

2.3 Word Alignment

The actual word alignment is based on the previously collected alignment candidates. Each bitext segment runs through a sequence of alignment steps. Candidates of word instance alignments can be compiled by associating possible link units. These units may include parts of the static segmentation or may be compiled dynamically. The same restrictions as in the candidate collection phase are applied in order to reduce the search space.

The alignment starts with the most reliable candidates. Each aligned token is removed from the text and only non-aligned tokens remain for the next step. In the current version of the system, 9 alignment steps are defined:

1. align one token units
2. align identical numerics
3. align cognates (string similarity scores, high threshold)
4. align strongly associated units (co-occurrence measures)
5. align low frequency pairs
6. align pairs from MRBDs
7. align cognates (string similarity scores)
8. align associated units (co-occurrence measures)
9. align remaining one token units

Each alignment step can be adjusted by several parameters such as LDR, link distances, string length and type. The alignment candidates are ranked by their probability (if an appropriate value is defined, e.g. Dice scores) and the most reliable pairs will be aligned first. Position weights can be used to modify probabilistic scores. The distance between the actual position and the estimated position of the aligned link unit, multiplied with a certain factor, is used to reduce the association score of each candidate pair. The reduction factor can be adjusted for each alignment step separately.

2.4 Iteration

Previously aligned word pairs that have been removed from the text can be added to the collection of alignment candidates (principle 6). The iteration process can be described as follows:

1. compile a bilingual lexicon from previously aligned words
2. compute alignment candidates by means of word association scores using the remaining tokens in the corpus
3. start the word alignment process all over again including an additional alignment step that applies the newly compiled lexicon
4. continue with (1) until no new alignments can be found (principle 8)

2.5 Evaluation

UWA stores information about each link. Each aligned unit is represented by a unique identifier corresponding to its origin in the bitext and its byte-span within the text relative to the beginning of the bitext segment. Reference data in form of a "gold standard" were manually defined for each bitext under consideration. UWA includes an evaluation module that compares results from a word alignment process with the gold standard. The module produces a protocol with information about each pair from the gold standard and summarizes the alignment result by counting the number of correct, partially correct, incorrect, and not aligned pairs. Finally, evaluation metrics are calculated. In this paper, $recall_{PWA}$, $precision_{PWA}$, and F-measure as their geometric mean were applied as proposed in Ahrenberg et al (forthcoming). Furthermore, information about the actual alignment step is stored for each aligned pair. In this way, the alignment process can be retraced and the quality of each step be investigated.

3. Experiments

UWA was tested with English/Swedish and Swedish/German bitexts from the PLUG corpus. For an illustration, word alignment results from an English→Swedish sub-corpus are presented here. The bitext is a literary text of some 130,000 words[4]. The gold standard comprises 500 random source language units that were linked manually using the Plug Link Annotator (Merkel et al forthcoming). In table 1, results of several word alignment experiments with different UWA settings are summarized.

	$precision_{PWA}$ (%)	$recall_{PWA}$ (%)	F (%)	time (min)[5]
static, no iteration	82.28	32.00	46.09	70
static	78.73	42.22	54.96	108
static+steps	79.88	43.91	56.67	115
semi-dynamic	77.34	43.05	55.32	131
semi-dynamic +steps	78.84	44.77	57.11	131
dynamic	78.73	43.80	56.28	250
dynamic +steps	78.27	44.29	56.57	261
parameter optimisation +MRBD	83.51	51.61	63.80	132

Table 1: Word alignment results for different UWA configurations.

The differences between each alignment configuration need to be explained. The first experiment represents the most basic configuration without iteration. The first three experiments apply *static* text segmentation only. Dynamic segmentation was applied in calculating string similarity measures and in the word alignment phase for the alignment

experiments that are denoted with *semi-dynamic*. The two *dynamic* approaches apply additionally dynamic text segmentation for the compilation of associated link candidates. The term *+steps* indicates the use of all alignment steps as described in section 2.3. The other experiments, except the last one, apply a simplified alignment procedure with one step for each candidate collection only. In the last approach, an empirically optimized UWA configuration and an additional basic dictionary were applied. Here, semi-dynamic text segmentation was used.

The results in table 1 confirm the practical use of the alignment principles that were described above. The approaches that apply a fine-tuned sequence of alignment steps (+step) contribute to the performance (in terms of F-values) at almost no expense (considering e.g. the processing time). Furthermore, the combination of static and dynamic text segmentation seems to be the most worthwhile approach. However, the set of alignment parameters has to be investigated further in order to discover potential improvements.

4. Conclusions

The Uppsala Word Aligner represents a highly adjustable word alignment system for fast and robust alignment of words and contiguous phrases from bilingual parallel texts. It supports different configurations and parameter settings for systematic investigations on translation units below the sentence level. The system applies knowledge-lite approaches that can be adjusted to different language pairs easily. Furthermore, supplementary modules and knowledge sources can be added. UWA is integrated in a modular corpus toolbox that provides convenient tools for experimentation, generation, and data organisation. It also includes a module for automatic evaluation using a previously defined gold standard. Thus, empirical investigations of different approaches and configurations are supported in a very efficient way.

UWA is implemented mainly in Perl and was tested on Linux. It will be available for academic research purposes at the end of the PLUG project.

[1] Knowledge-lite approaches in this case comprise techniques with minimal linguistic resources needed.

[2] The PLUG project is jointly funded by "The Swedish Council for Research in the Humanities and Social Sciences" HSFR and "The Swedish National Board for Industrial and Technical Development" NUTEK.

[3] The definite form of Swedish nouns is created by morphological modification.

[4] The word count includes both languages.

[5] The processing time includes phrase generation (8:42 min) and cognate extraction (23:54 min for static segmentation and 39:59 min for dynamic segmentation). However, these data are reusable and therefore, do not need to be compiled again for each alignment experiment.

References

Ahrenberg, L., Andersson, M. and Merkel, M. 1998. A simple hybrid aligner for generating lexical correspondences from parallel texts. In *Proceedings of COLING-ACL '98*, Montreal, Canada, pp. 29-35.

Ahrenberg, L., Merkel, M., Sågvall Hein, A., and Tiedemann, J. 1999. Evaluating LWA and UWA. PLUG deliverable 3A.1. Internal report.

Ahrenberg, L., Merkel, M., Sågvall Hein, A., and Tiedemann, J. forthcoming. Evaluation of Word Alignment Systems. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation, LREC-2000*, Athens, Greece, 2000.

Borin, L. 1998. Linguistics isn't always the answer: Word comparison in computational linguistics. In *Proceedings of the 11th Nordic Conference on Computational Linguistics NODALI98*, Center for Sprogteknologi and Department of General and Applied Linguistics, University of Copenhagen, pp. 140-151.

Dagan, I. and Church, K. W. 1994. Termight: Identifying and Translating Technical Terminology. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, Stuttgart/Germany.

van der Eijk, P. 1993. Automating the Acquisition of Bilingual Terminology. In *Proceedings of the 6th Conference of the European Chapter of the ACL*, 1993, Utrecht/The Netherlands.

Kay, M. 1980. The Proper Place of Men and Machines in Language Translation. Xerox PARC Working Paper, reprinted in *Machine Translation 12 (1-2)*, 1997, pp. 3-23

Melamed, I. D. 1995 Automatic Evaluation and Uniform Filter Cascades for Inducing N-best Translation Lexicons. In *Proceedings of the 3rd Workshop on Very Large Corpora*, Boston/Massachusetts.

Melamed, I. D. 1997. Automatic Discovery of Non-Compositional Compounds in Parallel Data. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP-2)*, Providence.

Melamed, I. D. 1999. Bitext Maps and Alignment via Pattern Recognition. *Computational Linguistics, 25(1)*, pp. 107-130.

Merkel, M., Andersson, M., and Ahrenberg, L. forthcoming. The PLUG Link Annotator - Interactive Construction of Data from Parallel Corpora. In *Proceedings from the Parallel Corpus Symposium*, April 22-23, 1999, Uppsala University.

Merkel, M. 1999. *Understanding and enhancing translation by parallel text processing.* Linköping Studies in Science and Technology. Dissertation No. 607. Linköping University. Dept. of Computer and Information Science.

Resnik, P. and Melamed, I. D. 1997. Semi-automatic acquisition of domain-specific translation lexicons. In *Proceedings of the Conference on Applied Natural Language Processing*, Washington, D.C.

Smadja, F., McKeown, K. R., Hatzivassiloglou, V. 1996. Translation Collocations for Bilingual Lexicons: A Statistical Approach. *Computational Linguistics, 22(1).*

Stephen, G. A. 1992. String Search. Technical report TR-92-gas-01, School of Electronic Engineering Science, University College of North Wales, Gwynnedd.

Sågvall Hein, A. forthcoming. The PLUG Project: Parallel corpora in Linköping, Uppsala, Göteborg: Aims and achievements. In *Proceedings from the Parallel Corpus Symposium*, April 22-23, 1999, Uppsala University.

Tiedemann, J. 1997. Automatical Lexicon Extraction from Aligned Bilingual Corpora. Diploma thesis, Otto-von-Guericke-University, Magdeburg, Department of Computer Science.

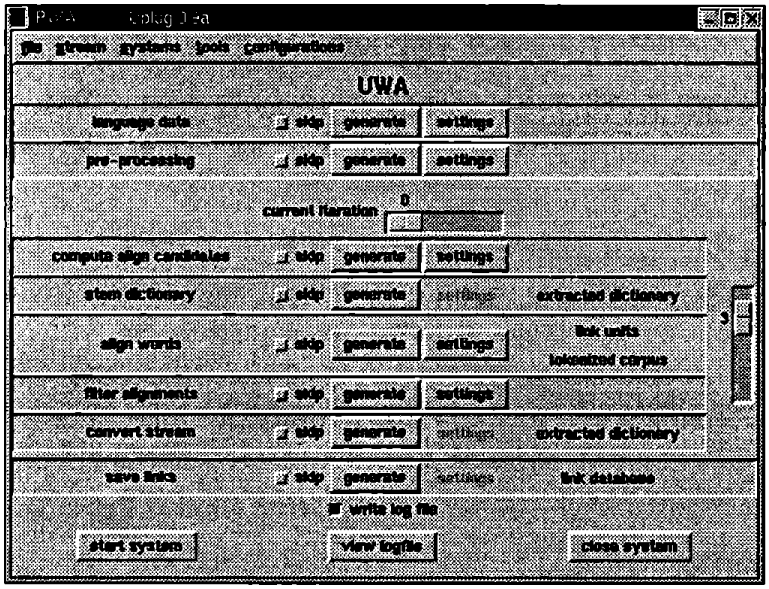
Tiedemann, J. 1998a, Extraction of translation equivalents from parallel corpora, In *Proceedings of the 11th Nordic Conference on Computational Linguistics NODALI98*, Center for Sprogteknologi and Department of General and Applied Linguistics, University of Copenhagen, pp. 120–128.

Tiedemann, J. 1998b. Parallell corpora in Linköping, Uppsala and Göteborg (PLUG). Work package 1. PLUG report, Department of linguistics, Uppsala university.

Tiedemann, J. 1999. Automatic Construction of Weighted String Similarity Measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Coropora*, University of Maryland, College Park/MD, pp. 213-219.

Tiedemann, J. forthcoming. Uplug - A Modular Corpus Tool for Parallel Corpora. In *Proceedings from the Parallel Corpus Symposium*, April 22-23, 1999, Uppsala University, Sweden.

Appendix A: UWA in the Uplug environment



Appendix B: MWU links from the English/Swedish test corpus (random sample)

source	target
Kippur War	Kippur-kriget
The Prime Minister	Premiärministerns
Kippur attack	Kippur-anfallet
Kosher food	Koschermat
Kultur paradise	Kultur-paradis
capitalist democracy	kapitalistiska demokratierna
careers be	karriärer
central tragedy	centrala tragedi
did mean	betydde
The Soviet Union	Sovjetunionen
anything	vad som helst
centralized state capitalism	centraliserad statskapitalism
The Times	New York Times
chamber music	kammarmusik
The United States	The United States
choke points	chokepunkterna
The West Bank	Västbanken
circus style	cirkusstil
The White House	Vita huset
citrus green	citrusgrönska
The Zionist movement	Sioniströrelsen
citrus groves	citruslundarna

On Using the Two-level Model as the Basis of Morphological Analysis and Synthesis of Estonian

Heli Uibo
Institute of Computer Science, University of Tartu, Estonia
heli_u@ut.ee

The paper deals with the problems of describing the Estonian morphological system in the two-level formalism, developed by Kimmo Koskenniemi. The outlines of Estonian morphology are drawn. The basics of the two-level model are given and illustrated with real examples from the experimental Estonian two-level morphology (**EETwoLM**) composed by the author. A detailed example of step-by-step morphological synthesis is given referring to all the relevant lexicons and rules. The compilation and testing processes using the XEROX finite-state software tools are described. Examples of morphological analysis and synthesis are demonstrated. The present stage of the system is characterised and the future perspectives are drawn. Finally, the suitability of the two-level model for the description of Estonian morphology is discussed.

1. Introduction

The module of morphological analysis and/or synthesis is unavoidable in any language engineering tool for Estonian because of its rich morphology. For example, in information retrieval systems, it is usually desirable to make queries using semantic entities, not using special morphological forms of a word. As the word stem often has several shapes in Estonian, the morphological component should belong to any information retrieval system.

Example 1. To make a query for all occurrences of the word "*rida*" ("row"), without a morphological synthesiser, three different queries are needed (we assume the possibility to add star (*) to the end of the stem instead of inflectional suffices):

*rida** (stem in strong grade)

*rea** (stem in weak grade)

ritta (singular additive ("to the row"), quite often used with this word)

The development of EETwoLM is not the first attempt to computerise the morphological analysis and synthesis of Estonian. Ülle Viks (1994) has done important research in the field of morphological classification of Estonian on the basis of pattern recognition theory starting from the 1970s. Viks (1992) has compiled the first morphological dictionary for Estonian as a practical output of the investigations. Further, E. Kuusik and Ü. Viks (1998) have implemented the rule-based morphological analysis and synthesis for Estonian. Heiki-Jaan Kaalep (1996) has developed the speller for Estonian using the results of Viks (1992).

Nevertheless, the growing popularity of the two-level model encourages us to consider its suitability to Estonian morphology. From the practical point of view, the appropriate description of Estonian morphology in the form of lexicons and two-level rules makes the significant move towards the application of Xerox language engineering tools to Estonian language (www.xerox.com/xrce/mltt).

2. The Brief Overview of Estonian Morphology

Estonian language is a member of Finno-Ugric family and is a close relative to Finnish. Estonian morphology is complex – inflected word-forms are built using both agglutination and stem flexion. Nouns have 14-15 cases. Plural forms often have parallel forms.

Table 1. Noun paradigm (word *"käsi"* ("hand")).

Case	Abbrev-iation	Singular		Plural	
		Word-form	**Meaning**	**Form**	**Meaning**
Nominative	N	*käsi*	(the) hand	*käed*	hands
Genitive	G	*käe*	of the hand	*käte*	of the hands

Partitive	P	*kätt*	hand (partial object)	*käsi* etc.
Illative	Ill	*käesse*	into the hand	*kätesse* or *käsisse*
Inessive	In	*käes*	in the hand	*kätes* or *käsis*
Elative	El	*käest*	out of the hand	*kätest* or *käsist*
Allative	All	*käele*	(on)to the hand	*kätele* or *käsile*
Adessive	Ad	*käel*	(up)on, at the hand	*kätel* or *käsil*
Ablative	Abl	*käelt*	from the hand	*kätelt* or *käsilt*
Translative	Trl	*käeks*	for, as the hand	*käteks* or *käsiks*
Terminative	Ter	*käeni*	up to, until the hand	*käteni*
Essive	Es	*käena*	as the hand	*kätena*
Abessive	Ab	*käeta*	without the hand	*käteta*
Comitative	Kom	*käega*	with the hand	*kätega*
Additive	Adt	*kätte*	(in)to the hand	-

Verbs have the following categories in Estonian: person (singular 1st to plural 3rd), voice (personal, impersonal), tense (present, imperfect, perfect, past perfect), mood (indicative, conditional, imperative, quotative).

Table 2. Part of the verb paradigm – infinite verb forms and indicative mood of the finite verb forms (word "*muutma*" ("to change")).

1. Infinite (declined) forms		
Morphological meaning	Abbreviation	Example
Supine (illative)	Sup	*muutma*
Supine inessive	Sup In	*muutmas*
Supine elative	Sup El	*muutmast*
Supine translative	Sup Tr	*muutmaks*
Supine abessive	Sup Ab	*muutmata*
Supine impersonal	Sup Ips	*muudetama*
Infinitive	Inf	*muuta*
Gerund	Ger	*muutes*

Participles (Pts):			
Present (Pr) participle	Personal (Ps)	Pts Pr Ps	*muutev*
	Impersonal (Ips)	Pts Pr Ips	*muudetav*
Past (Pt) participle	Personal	Pts Pt Ps	*muutnud*
	Impersonal	Pts Pt Ips	*muudetud*

2. Finite (conjugated) forms					
2.1. Indicative mood (Ind)					
Personal voice (Ps)					
Tense		Number	Person		
Present	Affirm-ation	Sg	1.	Ind Pr Ps Sg1	*muudan*
			2.	Ind Pr Ps Sg2	*muudad*
			3.	Ind Pr Ps Sg3	*muudab*
		Pl	1.	Ind Pr Ps Pl1	*muudame*
			2.	Ind Pr Ps Pl2	*muudate*
			3.	Ind Pr Ps Pl3	*muudavad*
	Negation (Neg)			Ind Pr Ps Neg	*ei muuda*

Imperfect	Affirm-ation	Sg	1.	Ind Ipt Ps Sg1	*muutsin*
			2.	Ind Ipt Ps Sg2	*muutsid*
			3.	Ind Ipt Ps Sg3	*muutis*
		Pl	1.	Ind Ipt Ps Pl1	*muutsime*
			2.	Ind Ipt Ps Pl2	*muutsite*
			3.	Ind Ipt Ps Pl3	*muutsid*
	Negation			Ind Ipt Ps Neg	*ei muutnud*
Present perfect	Affirmation			Ind Pf Ps	*on muutnud*
	Negation			Ind Pf Ps Neg	*ei ole muutnud*
Past perfect	Affirmation			Ind Ppf Ps	*oli muutnud*
	Negation			Ind Ppf Ps Neg	*ei olnud muutnud*
Impersonal voice (Ips)					
Present	Affirmation			Ind Pr Ips	*muudetakse*

	Negation	Ind Pr Ips Neg	*ei muudeta*
Imperfect	Affirmation	Ind Ipt Ips	*muudeti*
	Negation	Ind Ipt Ips Neg	*ei muudetud*
Present perfect	Affirmation	Ind Pf Ips	*on muudetud*
	Negation	Ind Pf Ips Neg	*ei ole muudetud*
Past perfect	Affirmation	Ind Ppf Ips	*oli muudetud*
	Negation	Ind Ppf Ips Neg	*ei olnud muudetud*

3. The Outlines of the Two-level Model, Illustrated with Examples in Estonian

The two-level morphology model was proposed by Kimmo Koskenniemi in his dissertation (1983). By now, the model has been used for morphological parsing of English, German, Swedish, Norwegian, Danish, Finnish, French, Turkish, Swahili etc. The main features of the two-level model are the following:

- The language description, consisting of rules and lexicons, is **separated** from the application programs.
- The model is **bidirectional** – it is oriented to morphological analysis as well as to morphological synthesis.
- **The two-levelness** of the model means that the deep representations of morphemes rather than morphemes themselves are maintained in lexicons. From those all the real word-forms can be produced with the help of two-level rules and links between lexicons.

Example 2. The lexical and surface representation of the word-form "*kaob*" ("disappears").

Lexical representation: k a D u $ + b

Surface representation: k a 0 o 0 0 b

The surface representation of a word-form is theoretically a sequence of phonemes. Practically, it tends to be the written form because of the better availability of written texts, as mentioned by Koskenniemi (1997:101).

The lexical representation can contain

a) surface phonemes ("k", "a", "u", "d", "b");
b) lexical phonemes ("D" corresponds to d in the strong grade and either disappears or assimilates in the weak grade);
c) special symbols for morpheme boundaries and morphophonological features ("+" indicates the boundary between stem and inflectional ending, "$" is the weak grade marker).

The representations are aligned with zero-characters.

- **Rules** and **lexicons** are two major parts of the model.
- **The set of rules** is like a filter, through which the lexical representation can be seen as surface representation and vice versa.
- The rules are **not ordered** and all of them have to be satisfied at the same time.
- Rules are implemented as **finite-state automata**.
- A finite-state automaton can be represented as a regular expression, thus the rules are coded as **regular expressions**.

Example 3. Example of a rule: "The disappearance of **D** in the weak grade"

D:0	⇔	SylBg Vow Vow: _ (StemFinVow:) %$:;	!*laud-laua*
		Vow Vow Liq _ StemFinVow %$:;	!*siirdama-siirata*
		[e | i: | u: | ü:] _ StemFinVow: %$: ;	!*vedama-vean, rida-rea*
		õ _ e %$: ;	!*õde-õe, põdeda-põen*
		[Cons - [r|s]] a _ u: %$: ;	!*kaduda-kaon*

The rule should be read as "The lexical symbol D corresponds to zero-character (i.e. disappears on the surface) in one of the following contexts and only there." In a context description the underline character "_" denotes the place of the pair **D:0** between the left and right contexts.

There is a possibility to define sets of characters to make the rules shorter and more readable, e.g. **Vow** stands for vowels, **StemFinVow** for possible stem final vowels, **Cons** for consonants. Sometimes it is also convenient to give names to frequent word segments, e.g. **SylBg** means the beginning of a syllable. Note that there is "$" (the weak grade marker) at the end of each right context, thus the disappearance takes place only in the weak grade.

The exclamation mark indicates the beginning of comments – we have provided every context with 1-2 example-words that help to understand the context.

- **The network of lexicons** consists of a stem lexicon and a number of small lexicons describing stem end alternations, inflectional and derivational processes.
- The network of lexicons is implemented as a **finite-state transducer**.
- A lexical entry includes **morphological information**, **lexical representation** and the name of the **next lexicon**.

Example 4. The structure and co-operation of lexicons.

```
LEXICON V11 !Conjugation class 1, endings which can be added to infinitive stem
+Inf:+da #;
+Ger:+des #;
+Kvt+Pr:+vat #;
+Ind+Pf:+nud #;
+Ind+Ipt: Ipt2;
+Ind+Imp+Pr: Imp;
LEXICON Ipt2 !The imperfect tense endings for the word types "elama" and "õppima"
 +Sg1:+sin #;
 +Sg2:+sid #;
 +Sg3:+s #;
 +Pl1:+sime #;
 +Pl2:+site #;
 +Pl3:+sid #;
LEXICON Imp  !The endings for imperative mood, except Sg2.
 +Sg1:+gu #;
 +Sg3:+gu #;
 +Pl1:+gem #;
 +Pl2:+ge #;
 +Pl3:+gu #;
```

Let us illustrate the details of information represented in the lexicons with two records:

+Inf	**:**	**+da**	**#**
+Ind+Imp+Pr	**:**		**Imp**
morphological information	separator	lexical representation	link to the next lexicon ('#' – the end of word-form)

If we take the word "*õppima*" ("to learn"), the LEXICON V11 builds the following forms:

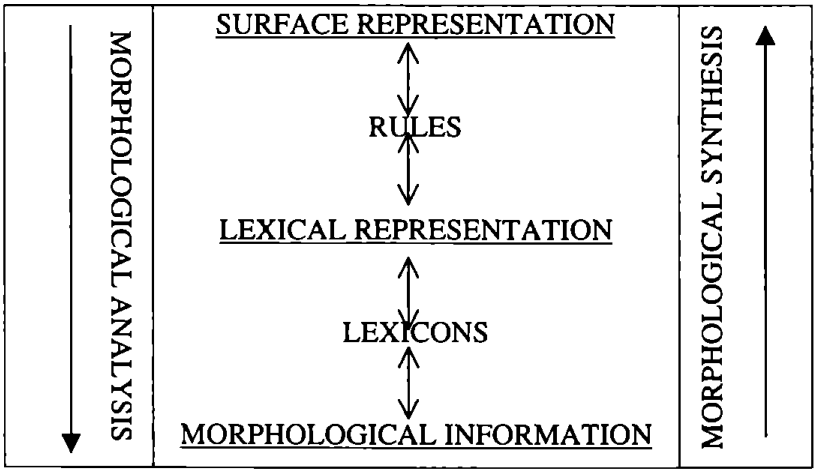
da-infinitive	- *õppida*
gerund	- *õppides*
quotative mood, present tense	- *õppivat*
indicative mood, present and past perfect tense	- *õppinud*

Further word-forms can be built going along the links to the lexicons Ipt2 and Imp:

indicative mood, imperfect tense	- *õppisin, õppisid, õppis, õppisime, õppisite*
imperative mood, present tense	- *õppigu, õppigem, õppige*

As have been said previously, the model can be the basis for morphological analysis as well as for synthesis. Both analysis and synthesis mean the sequential application of the rule automata and the lexical transducer, but in different order, as seen on figure 1.

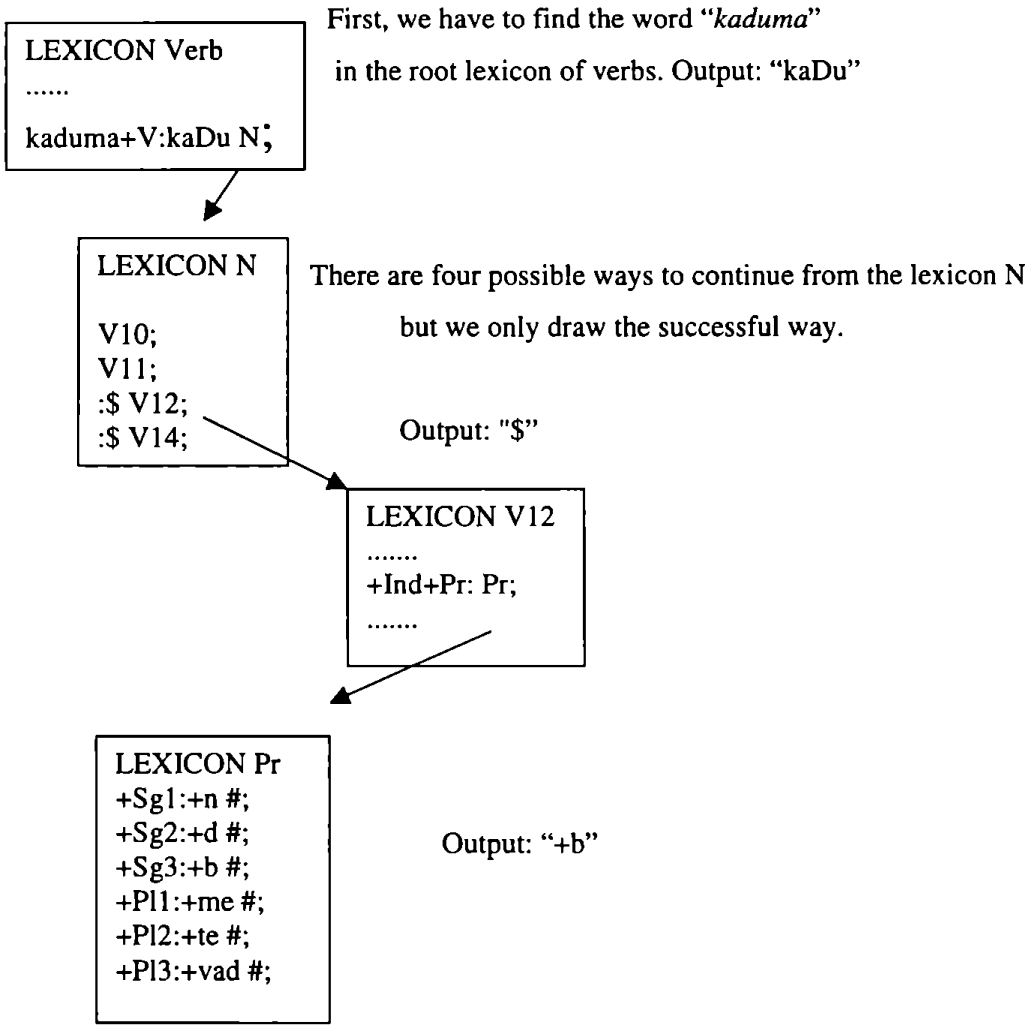
Figure 1. Morphological analysis and synthesis in the two-level model.



4. A Detailed Example of Morphological Synthesis

Example 5. The synthesis of the word-form "*kaob*" (verb "*kaduma*", indicative mood, present tense, singular, 3rd person).

Input: "kaduma+V+Ind+Pr+Sg3"



Thus, moving along the links between lexicons the **lexical representation "kaDu$+b"** has been composed.

Next, the rules will be applied. The rule "Disappearance of D" is satisfied with the pair D:0 in the context k a _ u: $: (see the last context). Thus we get "ka0u$+b".

```
Rule "Disappearance of D"
D:0 ⇔ SylBg Vow Vow: _ (StemFinVow:) %$:;
          Vow Vow Liq _ StemFinVow %$:;
          [ e | i: | u: | ü: ] _ StemFinVow: %$: ;
                        õ _ e %$:
             [Cons- [r|s] ] a _ u: %$: ;      !kaduda-kaob
```

```
Rule "Lowering of Vowels"
HighVow:LowVow ⇔ SylBg _ DCons: [aleli:lu:](l) %
        SylBg Vow DCons: _ %$: ;
            where HighVow in (u ü i)
                  LowVow in (o ö e)
            matched ;
```

The second rule accepts the pair u:o in
the context k a D: _ $: . The result is "ka0o$+b".

After applying the whole rule set the default pairs "$:0" and "+:0" have their turn. The result is "ka0o00b". After the deletion of zero-characters **the surface representation "kaob"** is ready.

5. The Implementation of the Two-level Model Using the XRCE Software

The rules and lexicons were developed and tested using the XEROX finite-state tools *lexc* (finite-state **lex**icon **c**ompiler developed by L. Karttunen (1993)) and *twolc* (**two**-level rule **c**ompiler developed by L. Karttunen & K. Beesley (1992)).

The process of testing the correctness and consistency of the lexicons and rules usually proceeds as follows:

- The rule and lexicon files have to be composed in a word processor in the described formats.
- Rules coded as regular expressions are compiled to automata with the two-level rule compiler "*twolc*". Lexicons are compiled into a lexical transducer with the lexicon compiler "*lexc*" .
- Next, the rules and lexicons can be composed with the help of the program "*lexc*".
- There are some possibilities to test the correctness of the language description in the program "*lexc*". One can analyse single word-forms using the directive "*lookup* <word-form>" and produce word-forms using the directive "*lookdown* <primary form+morphological information>"). We can use the directive "*random-surf*" for generating word-forms randomly using the existing lexicons and rules.

Example 6. Test of morphological analysis and synthesis using the program *lexc*.

lexc> lookup pead

pea+S+Sg+P ("head", substantive, singular partitive)

pea+S+Pl+N ("head", substantive, plural nominative)

pidama+V+Ind+Pr+Sg2 ("must", verb, indicative mood, present tense, singular, 2nd person)

As we can see, words can be morphologically ambiguous in Estonian. By the way, in Estonian texts about 50 % of the word-forms are morphologically ambiguous.

lexc> lookup ajalehepoisina

aeg+leht+poiss+S+Sg+Ess

In Estonian compound words the first components usually remain unchanged, while the last component is subject to inflection. At the same time, pre-components can be either in the nominative or in the genitive case. There is no general rule for choosing the right case for the pre-components.

lexc> lookdown kallis+A+Spr+Pl+El

kalleimaist

kalleimatest

The example demonstrates the existence of parallel forms, i.e. word-forms having the same grammatical meaning but different forms.

Example 7. Random surface string generation, using the program *lexc*.

lexc> random-surf

Use (s)ource or (r)esult? [r]:

NOTE: Using RESULT.

käed	+	"hands"
pessa	+	"into the nest"
öeldavaid	+	"said by somebody" (Adj), plural partitive
kohaehe	?	"the decoration of a place" – strange compound
vanalt	+	"from the old" (Adj)

pimeduse	+	"of the darkness"
ülejätnuiks	?	strange compound, hard to translate
eemaltõppijad	?	"ones who learn from a distance" – slightly strange compound
nähtuta	+	"without the seen thing"
läksin	+	(I) "went"

To the output of the program approximate English translation as well as signs "+" or "?" are added. Every normal word-form is followed by "+". If the word-form is not used, it is marked by "?". The mistakes are caused by the overgeneration of compounds and derivatives.

6. Results

The experimental two-level morphology for Estonian (EETwoLM) has been composed:

- There are 45 two-level rules in the rule set that deal with stem flexion, phonotactics, orthography and morphophonological distribution.
- The net of lexicons consists of root lexicons for all word classes containing a total of ≈350 different word roots and of over 200 small lexicons describing the stem end alternations, conjugation of verbs and declination of nouns.
- The lexicons and rules express most of the phenomena occurring in Estonian morphology.
- The system is consistent in its present stage: we can get correct results to both morphological analysis and synthesis in the range of word stems occurring in the root lexicons.

7. Future Perspectives

The coverage of stem lexicons can be enlarged semi-automatically, using the electronic version of Viks (1992) and the type-detection module developed in the Institute of Estonian Language (see the webpage *www.eki.ee/tarkvara*). To adapt EETwoLM exactly to the morphological classification after Viks (1994), some changes have to be introduced into the network of lexicons.

A consistent and lexically satisfactory description of Estonian morphology in the two-level formalism can be the basis of automatic morphological analysis and synthesis. Simultaneously, two-level-morphology-based language engineering software in XRCE (spelling checker, information retriever a. o) would be applicable to Estonian language.

8. The Estimation of the Suitability of the Two-level Morphology Model to Estonian Language

During the composition of EETwoLM some features of the two-level model proved very useful. We have given the overview of them in Uibo (1999:55):

1. Using the lexical representation is an advantage because the lexical entries can include other information additionally to the pure sequence of letters:
   - There is a possibility to use special denotations for phonemes having more than one surface variant. This is a great advantage, as the type of stem flexion generally does not depend on the phonemic shape of the stem in the present-day Estonian - some kinds of stem flexion are not productive any more.
   - The lexical information can contain morphophonological features and morpheme boundaries, which are often used by rules.
2. The rule set is not ordered. The compilation of an ordered rule set would be complicated because it is difficult to count the influence of all the previous rules in the sequence to the left and right context.
3. A rule can point to the arbitrarily far context. E.g. there can be a rule which should check the stem final character, without checking the number of syllables.
4. If a pair occurs in several contexts having nothing common neither in content nor in form the corresponding contexts can be listed on the right side of one and the same rule. It prevents from introducing new and meaningless lexical characters. E.g. the pair "S:0" is possible both in the weak grade of the words with s:0-alternation within the stem and at the end of a class of words ending with "s". In the first case the lexical phoneme "S" is situated in between vowels, in the second case it is found at the end of the stem.
5. The net of lexicons is convenient to handle
   - non-phonologically caused stem end alternations (org. "ne/se", "0/me");

- rules of morphotactics;
- productive derivation and compounding (partly).

However, we have also pointed to some difficulties in Uibo (1999:56) that have occurred in the course of the description of Estonian morphology in the two-level formalism:

1. The word class is subject to change during the derivation processes, but the morphological information is composed moving along the pointers between lexicons in one direction. Return to the previous steps, thus the deletion and replacement of the word class is not possible. Now the problem has been solved artificially: the verb derivatives are in a separate lexicon and for the productively derivable adjectives the determination of word class has been deferred.
2. It is inconvenient to introduce word lists into the lexicon system that do not coincide with the inflection types. The lists are needed e.g. for words with exceptional forms, for words having additive case and short plural, and especially for compound word production.

The hypothetical solution of the listed problems could be the combination of the two-level model with another model that would help to overcome the above-listed limitations.

9. Conclusion

The experiments on EETwoLM have shown that the two-level model is quite usable for Estonian simple word recognition and production. However, the net of lexicons is not very well suitable for modelling the derivation and compounding processes. The efficiency of the implementation of the rules and lexicons as finite-state transducers is definitely an advantage. Unfortunately, the objective evaluation of EETwoLM is not possible yet, as the coverage of the lexicons is insufficient for real text processing.

References

Kaalep, H.-J. 1996. ESTMORF, a Morphological Analyser for Estonian. *Estonian in the Changing World* / edited by H. Õim, 43-97. Tartu: University of Tartu, Dept of General Linguistics.

Karttunen, L. & Beesley, K. R. 1992. Two-level Rule Compiler. Technical Report. ISTL-92-2. October 1992. Palo Alto, California: Xerox Palo Alto Research Centre.

Karttunen, L. 1993. *Finite-State Lexicon Compiler.* Technical Report. ISTL-NLTT-1993-04-02. April 1993. Palo Alto, California: Xerox Palo Alto Research Centre.

Koskenniemi, K. 1983. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production.* Helsinki: University of Helsinki, Dept of General Linguistics. Publications No. 11.

Koskenniemi, K. 1997. Representations and Finite-State Components in Natural Language. *Finite-state Language Processing* / edited by E. Roche and Y. Schabes, 99-116. Language, Speech, and Communication Series. Cambridge, Massachusetts, London, England: The MIT Press.

Kuusik, E. & Viks, Ü. 1998. The Rule-based Morphological Synthesis. *Arvutimaailm* (*The World of Computers*) 1/1998, 43-45, 63, 2/1998, 19-21 (in Estonian).

Uibo, H. 1999. The Estonian Word-Form Analysis and Generation, Using Two-Level Morphology Model. M.Sc. thesis. Tartu: University of Tartu, Institute of Computer Science (in Estonian).

Viks, Ü. 1992. *A Concise Morphological Dictionary of Estonian.* Tallinn: Institute of Estonian Language and Literature.

Viks, Ü. 1994. Classificatory Morphology. Ph.D. thesis. Tartu: University of Tartu. (in Estonian).

LFG-DOT: Combining Constraint-Based and Empirical Methodologies for Robust MT

Andy Way,
School of Computer Applications,
Dublin City University.

Email: away@compapp.dcu.ie

Abstract

The Data-Oriented Parsing Model (DOP, [1]; [2]) has been presented as a promising paradigm for NLP. It has also been used as a basis for Machine Translation (MT) — Data-Oriented Translation (DOT, [9]). Lexical Functional Grammar (LFG, [5]) has also been used for MT ([6]). LFG has recently been allied to DOP to produce a new LFG-DOP model ([3]) which improves the robustness of LFG. We summarize the DOT model of translation as well as the DOP model on which it is based. We demonstrate that DOT is not guaranteed to produce the correct translation, despite provably deriving the most probable translation. Finally, we propose a novel hybrid model for MT based on LFG-DOP which promises to improve upon DOT, as well as the pure LFG-based translation model.

1 Introduction

Neither of the main paradigmatic approaches to MT, namely rule-based and statistical, currently suffice to the standard required. Nevertheless, each contains elements which if properly harnessed should lead to an overall improvement in translation performance. It is in this new hybrid spirit that our search for a better solution to the problems of MT can be

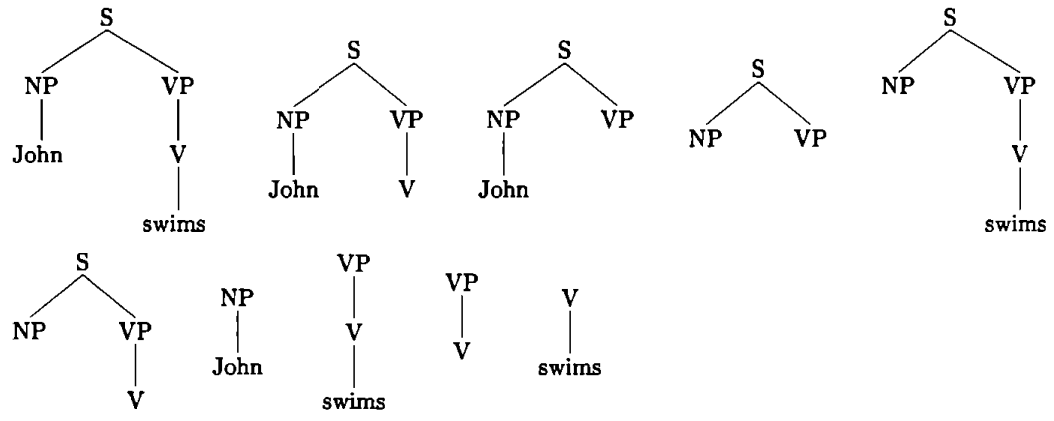
seen. We propose that combining DOP ([1];[2]) with the conventional transfer rules of LFG ([6]) may derive a new model for MT, LFG-DOT, which promises to improve upon DOT, as well as the pure LFG-based translation model.

2 The DOP Architecture for NLP

DOP language models ([1];[2]) assume that past experiences of language are significant in both perception and production. DOP prefers performance models over competence grammars, in that abstract grammar rules are eschewed in favour of models based on large collections of previously occurring fragments of language. New language fragments are processed with reference to already existing fragments from the corpus, which are combined using probabilistic techniques to determine the most likely analysis for the new fragment.

DOP models typically use surface PS-trees as the chosen **representation** for strings (hence "Tree-DOP"), but nothing hangs on this choice. However, given that LFG c-structures are little more than annotated PS-trees allows us to proceed very much on the same lines as in Tree-DOP, which has two **decomposition** operations to produce subtrees from sentence representations: (i) the *Root* operation, which takes any node in a tree as the root of a new subtree, deleting all other nodes except this new root and all nodes dominated by it; and (ii) the *Frontier* operation, which selects a (possibly empty) set of nodes in the newly created subtree, excluding the root, and deletes all subtrees dominated by these selected nodes.

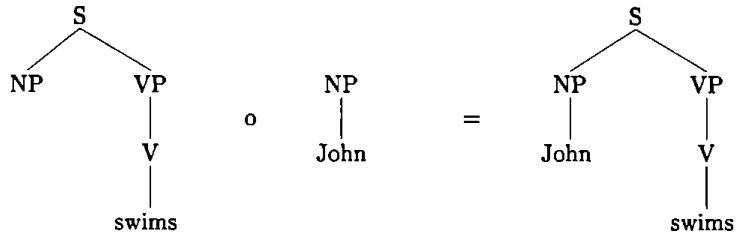
(1)



The full set of DOP trees derived from the sentence *John swims* are those in (1).

Tree-DOP **recombines** fragments starting from the leftmost non-terminal frontier node, and replaces this with a fragment having the same root symbol. For instance, assuming the treebank in (1), *John swims* has (2) as a possible derivation (among many others):

(2)



Finally, the chosen **probability model** for Tree-DOP is based quite simply on the relative frequencies of fragments in the corpus.

These elements enable representations of new strings to be constructed from previously occurring fragments in a number of ways. If each derivation t has a probability P(t) (i.e. its relative frequency), then the probability of deriving a Tree-DOP representation is the sum of the probabilities of the individual derivations, as in (3):

(3)

$$\sum i \prod j \frac{\#(t_{ij})}{\#(t \mid root(t) = root(t_{ij}))}$$

The probability of each individual derivation t is calculated as the product of the probabilities of all the constituent elements $\langle t_1, t_2 ... t_n \rangle$ involved in choosing tree t from the corpus, as in (4):

(4)

$$P(\langle t_1, t_2 ... t_n \rangle) = \prod_{i=1}^{n} \frac{P(t)}{\sum_{t' \in corpus} P(t')}$$

Given these formulae, the probability of the derivation for *John swims* in (2) is $\frac{1}{6}$. This is calculated by multiplying together the probability of each of the two tree fragments involved in the derivation, namely those in (5):

(5) P(t = [**NP vp[v[swims]]**] | root(t) = S).P(t = [**np[John]**]| root(t) = NP) = $\frac{1}{6} \cdot \frac{1}{1} = \frac{1}{6}$

The probability of the *parse* of *John swims*, however, is calculated by summing all derivations resulting in the parse-tree for the sentence (as (3) shows), which, given the trivial corpus in

(1), is 1. However, adding the fragments from a new sentence *Peter laughs* to the treebank in (1) allows us now to derive the probability of two new strings – *Peter swims* and *John laughs* – with respect to this small corpus of tree fragments. In this way, it can be seen that DOP handles unseen data on the basis of previous experience – despite the fact that we have never seen either new sentence before, we are able to process them compositionally, on the basis of previously occurring fragments of each in our corpus. Each tree which can play a part in combining together with other trees to form a representation for a sentence is used to contribute to the overall probability of that representation given the corpus.

2.1 Opportunities for Hybridity—LFG DOP

DOP-based approaches are necessarily limited to those contextual dependencies actually occurring in the corpus, which is a reflection of surface phenomena only. Given its facility to capture and provide representations of linguistic phenomena other than those occurring at surface structure, the functional structures of LFG have been allied to the techniques of DOP to create a new model, LFG-DOP ([3]), which adds a measure of robustness not available to models based solely on LFG. We suggest that this framework has the potential to be utilised for MT.

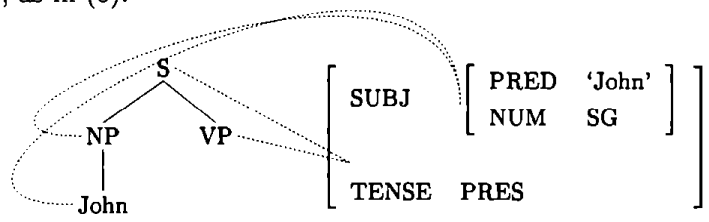
As with DOP, LFG-DOP needs to be defined using four parameters. Its **representations** are simply lifted *en bloc* from LFG theory, so that each string is annotated with a c-structure, an f-structure, and a mapping ϕ between them, with well-formedness conditions operating solely on f-structure, as usual.

Since we are now dealing with ⟨c,f⟩ pairs of structure, the *Root* and *Frontier* **decomposition** operations of DOP need to be adapted to stipulate exactly which c-structure nodes are linked to which f-structure components, thereby maintaining the fundamentals of c- and f-structure correspondence. As in DOP, *Root* erases all nodes outside of the selected node, except this new root and all nodes dominated by it, and in addition deletes all ϕ-links leaving the erased nodes, as well as all f-structure units that are not ϕ-accessible from the remaining nodes, reflecting the intuitive notion that nodes in a tree carry information only about the f-structure elements to which the root node of the tree permits access.

Frontier operates as in DOP, selecting a set of nodes in the newly created subtree, excluding the root, and deleting all subtrees dominated by these selected nodes. Furthermore, it deletes all ϕ-links of these erased nodes together with any semantic form corresponding to the same

nodes, as in (6):

(6)



which illustrates the ability of *Root* nodes to access certain features (TENSE, here) even after subnodes have been deleted. (6) can be pruned still further by applying a third, and new operation, *Discard*, to the TENSE feature. *Discard* adds considerably to LFG's robustness by providing generalised fragments from those derived via *Root* and *Frontier*.

Composition is also a two-step operation. C-structures are combined by left-most substitution, as in DOP, subject to the matching of their nodes. F-structures corresponding to these nodes are then recursively unified, and the resulting f-structures are subjected to the grammaticality checks of LFG.

Finally, $P(f \mid CS)$ denotes the probability of choosing a fragment f from a competition set CS of competing fragments. [3] describe four possible competition sets linked to the **probability models** for LFG-DOP: (i) a straightforward extension of the DOP probability model where the choice of a fragment depends only on its *Root* node and not on the Grammaticality conditions of LFG; (ii) c-structure nodes must match, and f-structures must be unifiable if two LFG fragments are to be combined, i.e. taking into account the LFG Uniqueness condition as well as the *Root* category; (iii) furthermore, the LFG Coherence check is enforced at each step; and (iv) finally, all LFG grammaticality checks, as well as the DOP category-matching stipulation, are left to the end. Note that in models (i)–(iii) the category matching condition is enforced on-line whilst all LFG checks are either performed on-line or *post hoc*, whereas given the non-monotonic nature of the Completeness check, this can only ever be enforced *post hoc*.

3 Data-Oriented Translation (DOT)

[9] has developed a DOP-based model of translation — Data-Oriented Translation — which relates POS-fragments between two languages (English and Dutch), with an accompanying probability. Once a derivation for the source language sentence has been arrived at, the target structure is assembled, and a string produced. Since there are typically many different

derivations for the source sentence, there may be as many different translations available. As is the case when DOP is used monolingually, the probability of a translation is calculated by summing the probabilities of all possible derivations forming the translation. Poutsma shows that the most probable translation can be computed using Monte-Carlo disambiguation, and exemplifies this using sentence idioms, where corresponding source-target translations are linked at all possible nodes.

3.1 Some Limitations of DOT

DOT is an interesting model, yet it fails to capture the correct translation when this is non-compositional and considerably less probable than the default, compositional alternative. When LFG-DOP MT (LFG-DOT) is used instead this problem may be overcome. Furthermore, DOP's statistical model also gives a "level of correctness" figure to alternative translations. This is useful in cases where the default translation in LFG-MT (and in many other systems) cannot be suppressed when the specific translation is required. For example, assuming the basic default rules in (7):

(7) a. commettre ⇔ commit

b. suicide ⇔ suicide

in order to deal with the sentences in (8):

(8) a. Jean commet un crime ⇔ Jean commits a crime

b. Le suicide est tragique ⇔ Suicide is tragic

we would get the wrong translation where *John commits suicide* ⇔ **John commet le suicide* (cf. *John se suicide*). We would like specific rules to override the default translation where applicable, but this is not possible in LFG-MT, so we would get both translations here, i.e. a correct one (via the specific τ-equations in (10)) and a wrong one (via the default τ-equations, required to translate *commettre* as *commit* in other circumstances). Assuming a DOP treebank built from the French sentences in (8) as well as *Marie se suicide*, the ill-formed string *Jean commet le suicide* is preferred (in the French language model) about half as much again as the correct alternative *Jean se suicide*. There are several reasons for this: the preference for *Jean* as subject of *commettre*, the co-occurrence of *le* and *suicide*, plus the fact that *commettre* is followed by an NP consisting of a Det + N sequence. Note

also that these results are obtained with the same number of instances of each verb — in a larger corpus *commettre* would surely greatly outnumber instances of *se suicider*.

This is by no means an unexpected result. As an example, in the *LOB Corpus*, there are 66 instances of *commit* as a verb (including its morphological variants), only 4 of which have *suicide* as its object, out of the 15 occurrences of *suicide* as an NP. Consequently, even for this small sample, we can see that 94% of these examples need to be translated compositionally (by *commettre* + NP), while only the *commit suicide* examples require a specific rule to apply (i.e. *se suicider*). In the on-line *Canadian Hansards* covering 1986-1993, there are just 106 instances of *se suicider* (including its morphological variants). There will, of course, be many thousands of instances of *commettre*. Given occurrences of *suicide* as an NP in French corpora, it is not an unreasonable hypothesis to expect that the wrong, compositional translations will be much more probable than those derived via the specific rule.

Given Poutsma's model, it would appear that the adherence to left-most substitution in the target given *a priori* left-most substitution in the source is too strictly linked to linear order of words, so that, as soon as this deviates to any significant extent even between similar languages, DOT has a huge bias in favour of the incorrect translation. Even if the correct, non-compositional translation is achievable in such circumstances via DOT, it is likely to be so outranked by other wrong alternatives that it will be dismissed, unless all possible translations are maintained for later scrutiny by the user.

4 LFG-DOT: A New Theory of Translation

The DOT model cannot explicitly relate parts of the source language structure to the corresponding, correct parts in the target structure. One line of investigation which we now develop that can overcome this linear restriction is to use LFG-DOP ([3]) as the basis for an innovative MT system, using LFG's τ-equations to relate translation fragments between languages.

4.1 Model 1: $\langle c, \phi, f, \tau, f' \rangle$

Using separate language corpora, this simple, linear model builds a target f-structure f' from a source c-structure c and f-structure f, the mapping between them ϕ, and the tau-equations τ. From this target f-structure f', a target string is generated via the standard LFG generation algorithms ([7]; [11]). The probability of the target f-structure R_t being the translation of the source string W_s is:

$$(9) \qquad \begin{aligned} \max_{R_t} P(R_t \mid W_s) &= \max_{R_t} \sum_{R_{t,s}} P(R_s \mid W_s).P(R_t \mid R_s, W_s) \\ &= \max_{R_t} \sum_{R_{t,s}} P(R_s \mid W_s).P(R_t \mid R_s) \end{aligned}$$

incorporating a Markov assumption that the target f-structure's derivation from a source string (via ϕ and τ) is independent of the original words involved: it is dependent *solely* on the monolingual LFG-DOP representation assigned. This is an attempt to avoid as much as possible the sparse data problem, given that in all probability we will never have enough LFG-DOP fragments to model these numbers with any reasonable accuracy. The components needed given (9), therefore, are (i) a source language LFG-DOP model, $P(R_s \mid W_s)$; (ii) the τ mapping (the translation model) plus the associated probabilities that a source f-structure produces a target equivalent, $P(R_t \mid R_s)$.

The advantage of this model over DOT is the availability of the explicit τ-equations to link source-target correspondences, as in (10):

(10) *commit*: ($\tau\uparrow$ PRED) = se suicider, τ(↑ SUBJ) = ($\tau\uparrow$ SUBJ), (↑ OBJ PRED) $=_c$ suicide

Using LFG τ-equations ensures the derivation of the correct target f-structure, along with some wrong alternatives (here) via the default rules. We cannot be sure that the generation of a target string via the correct target f-structure will be a more probable translation than any wrong alternative, but it will exist as one of a small number of high-ranking candidate solutions from which the final translation can be selected. Of course, we may instead choose to derive the target string using a target language LFG-DOP model (via ϕ') rather than the standard LFG generation algorithms, in which case the probability model in (9) needs to be adapted to incorporate $P(W_t \mid R_t)$, where again we presume that the target string generation is independent of all source language representations: it is dependent solely on the τ-equations derived from the source f-structure.

4.2 Model 2: $\langle c, f, \phi \rangle \longrightarrow \gamma, \tau \longleftarrow \langle c', f', \phi' \rangle$

Here we have integrated language corpora, where for each node in a tree c, we relate it both to its corresponding f-structure fragment f and its corresponding target c-structure node c', and for each source f-structure fragment, we relate that to its target language fragment in f-structure f', via τ. The probability model used this time is:

$$(11) \qquad \overset{Max}{t} P(t \mid s) = \overset{Max}{t} \sum_{R_{t,s}} P(t, R_{t,s} \mid s) = \overset{Max}{t} \sum_{R_{t,s}} P(R_t \mid s)$$

where now $R_{t,s}$ are the full $\langle c, f \rangle$ representation pairings for the target and source strings, respectively. Our basic units are pairs of linked LFG-DOP fragments (cf. the linked DOP fragments in DOT, [9]), and the basic stochastic event is the combination of two linked LFG-DOP fragment pairs. Thus, we compute the probability of $P(t \mid s)$ by the sum of the probabilities of all R_t, R_s pairs that generate t and s (and, ultimately, of course, choosing that t for which this probability sum is maximal), where the probability of an R_t, R_s pair is computed as the sum of the probabilities of its derivation-pairs; each derivation-pair is the product of its linked fragment-pairs; and each linked fragment-pair has a probability equal to its normalized relative frequency. Bod & Kaplan discuss four different ways of calculating the probability of an (unlinked) fragment, depending on which LFG grammaticality checks (if any) are integrated into the competition sets assumed (cf. section 2.1).

The principal reason for hypothesising the γ function in this model is that it is reasonable to assume, as [9] has shown, that valuable information concerning the final formulation of the target string can be influenced by the source c-structure. In this way we have two pieces of information at hand with which to build the target string—the γ and ϕ' functions, which if they can be properly harnessed, should bring about a better translation, given the extra evidence that is being brought to bear in its generation.

4.3 Semi-Automatic Creation of LFG & LFG-DOP Corpora

A major problem for researchers interested in LFG and LFG-DOP is the absence of suitable, extensive corpora. Given this, in order to demonstrate practically the feasibility of LFG-DOT, we have begun to develop our own LFG and LFG-DOP corpora ([10]).

Initially we took the publicly available set of 100 sentences of the *AP Treebank* ([8]). Despite its small size, this was sufficiently large to demonstrate the plausibility of our approach. One

particular entry is:

(12)
```
A001  39 v
[N The_AT march_NN1 N][V was_VBDZ [J peaceful_JJ J]V] ._.
```

We then automatically extract the rules from this corpus (following the method of [4]), and create automatically LFG-macros for each lexical category:

(13)
```
macro(at(Word),FStr) :-
  FStr:spec === Word.

macro(jj(Word),FStr) :-
  FStr:pred === Word.

macro(nn1(Word),FStr) :-
  FStr:pred === Word,
  FStr:num  === sg.

macro(vbdz(_Word),FStr) :-
  FStr:tense  === past,
  FStr:pred   === be.
```

We then annotate the extracted rules with LFG functional schemata by hand:

(14)
```
rule(n(A), [at(B),nn1(C)]) :-
   A === B,
   A === C.

rule(v(A), [vbdz(B),j(C)]) :-
   A === B,
   B:subj === C:subj,
   A:xcomp === C.

rule(j(A), [jj(B)]) :-
   A === B.

rule(sent(A), [n(B),v(C)]) :-
   A:subj === B,
   A      === C.
```

and 'reparse' the original treebank entries, *not* the strings, simply by recursively following the tree annotations provided by the original annotators. In so doing the interpreter solves the constraint equations associated with the grammar rules and lexical macros involved in the parse, returning single f-structures, as in:

(15)
```
subj : spec : the
       pred : march
       num  : sg
xcomp : pred : peaceful
        subj : spec : the
               pred : march
               num  : sg
tense : past
pred  : be
```

In order to produce target f-structures, all that is necessary is to add τ-equations to the

lexical and structural rules, and reparse the treebank entries. Once these target f-structures exist, we can test out the translation models and report results.

5 Conclusions

The DOT translation system, despite provably deriving the most probable translation, is not guaranteed to produce the best, or even a correct translation, since it is unable to explicitly link exactly those fragments which are playing the decisive role in translation.

[3] have shown how DOP and LFG can be integrated to provide a powerful mechanism for the treatment of parsing. We described how such a model may be extended to provide a robust solution for the problems of MT in the spirit of the current trend for hybrid approaches. LFG-DOT promises to improve on previous attempts at LFG-MT, particular where robustness is concerned, being able to handle both unseen and ill-formed input with relative ease. It also ensures that the correct target f-structure is input into the generation process. It is reasonable to expect LFG-DOT to outperform pure statistics-based systems, in having the additional facility of grammatical information at hand to use where necessary.

Much of this work is ongoing, and a number of issues remain for the future, especially the automatic creation of large LFG-DOP corpora necessary as training and test data for the translation models. This will complete the development of the systems described, leading to greater experimentation on a larger scale.

References

[1] Bod, R. (1995): *Enriching Linguistics with Statistics: Performance Models of Natural Language*, ILLC Dissertation Series 1995-14, University of Amsterdam, The Netherlands.

[2] Bod, R. (1998): *Beyond Grammar: An Experience-Based Theory of Language*, CSLI Publications, Stanford, California.

[3] Bod, R. & R. Kaplan (1998): "A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis", in COLING: *Proceedings of the 17th International Conference on Computational Linguistics & 36th Conference of the Association for Computational Linguistics*, Montreal, Canada, 1:145–151.

[4] Charniak, E. (1996): "Tree-bank grammars", in AAAI-96, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, MIT Press, pp.1031-1036.

[5] Kaplan, R. & J. Bresnan, (1982): "Lexical Functional Grammar: A Formal System for Grammatical for Grammatical Representation", in J. Bresnan (ed.) *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Mass., pp.173–281.

[6] Kaplan, R., K. Netter, J. Wedekind & A. Zaenen (1989): "Translation by Structural Correspondences", in *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, pp.272–281.

[7] Kohl, D. (1992): "Generation from Under- and Overspecified Structures", in COLING: *14th International Conference on Computational Linguistics*, Nantes, France, pp.686–692.

[8] Leech, G. and R. Garside (1991): "Running a Grammar Factory: on the Compilation of Parsed Corpora, or 'Treebanks' ", in: S. Johansson and A.-B. Stenström (eds), *English Computer Corpora: Selected Papers and Research Guide*, Mouton de Gruyter, Berlin, pp.15–32.

[9] Poutsma, A. (1998): "Data-Oriented Translation", in *Ninth Conference of Computational Linguistics In the Netherlands*, Leuven, Belgium.

[10] Van Genabith, J., A. Way and L. Sadler (1999): "Semi-Automatic Generation of F-Structures from Treebanks", in *Proceedings of LFG-99*, Manchester, UK.

[11] Wedekind, J. (1988): "Generation as Structure Driven Derivation", in COLING: *12th International Conference on Computational Linguistics* , Budapest, Hungary, pp.732–737.