# Proceedings of the Sixteenth Computational Linguistics in the Netherlands

**Edited by:**
**Khalil Sima'an,**
**Maarten de Rijke,**
**Remko Scha and**
**Rob van Son**

**Universiteit van Amsterdam**

# The Sixteenth Computational Linguistics in the Netherlands

16 December 2005
Amsterdam

**Organizing Committee**

Maarten de Rijke, Remko Scha, Rob van Son and Khalil Sima'an

FACULTY OF SCIENCE FACULTY OF HUMANITIES

UNIVERSITEIT VAN AMSTERDAM

Printed by Grafisch Centrum Amsterdam (GCA)

**ISBN-10 90-5776-161-0**
**ISBN-13 978-90-5776-161-4**

# Contents

# Preface

KHALIL SIMA'AN

*Institute for Logic, Language and Computation*
*University of Amsterdam*

This volume presents selected contributions to the 16th Meeting of Computational Linguistics in the Netherlands (CLIN 2005), held at the University of Amsterdam on December 16, 2005. The CLIN 2005 meeting consisted of 42 presentations (selected from more than 50 submitted abstracts) and was attened by more than 110 participants. Perhaps the highlight of the meeting was the two invited talks by Hermann Ney (RWTH, Aachen, Germany) and Eduard Hovy (ISI, University of Southern California, USA). As other preceding CLIN meetings, the 2005 meeting remained a forum for presenting diverse work concerning general computational formalisms, techniques, models and applications that concern language and speech processing.

For the call for papers that followed the 2005 meeting, there were in total 13 submissions of which we selected 8 for this volume. The articles found in this volume can be considered a sample from the distribution of current activity within the Netherlands and Belguim in building corpora and tools for processing the Dutch language, and in studying general computational linguistic topics such as parsing, learning and algorithms. It cannot escape the eye of the observer that much of the current computational linguistics research within the Dutch language area is funded by research programmes aimed at improving the position of the small Dutch language within Europe. While we support this goal wholeheartedly, we have aimed at keeping the CLIN meeting up to its scientific tradition with regard to the scientific orientation, the invited speakers and the computational flavour.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands.*
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

**Organizing committee:** The CLIN 2005 meeting was organized by Maarten de Rijke, Remko Scha, Rob van Son and Khalil Sima'an. Valuable help on the day itself was provided by Krisztian Balog, Aspasia Beneti, Fransje Enserink, Felix Hageloh, Joeri Honnef, Valentin Jijkoun, Tanja Kassenaar, Markos Mylonakis, Irwin Oppenheim, Jessica Pogorzelski, Detlef Prescher, Koen van de Sande, Yoav Seginer, Erik Tjong Kim Sang, Reut Tsarfati, Roberto Valenti, Marjan Veldhuizen, Klara Weiand, Wieneke Wesseling, Jelle Zuidema, and Janneke van der Zwaan.

**Reviewing of papers:** For the present proceedings we were supported by a set of capable reviewers who provided us and the authors with detailed reviews: Rens Bod, Antal van den Bosch, Gosse Bouma, Walter Daelemans, Frank van Einde, David Ahn, Henk van den Heuvel, Mary Hearne, Veronique Hoste, Jaap Kamps, Mark-Jan Nederhof, Gertjan van Noord, Paola Monachesi, Guy de Pauw, Detlef Prescher, Rob van Son, Eric Tjong Kim Sang, Ton van der Wouden, Henk Zeevat and Jelle Zuidema. We thank them all for their dedicated work. They made the task of selecting the articles easier and improved the overal quality of the proceedings.

**Sponsors:** The CLIN 2005 meeting was generously sponsored by the following organizations:

**ACLC** Amsterdam Center for Language and Communication (UvA),
**FNWI** Faculty of Science (UvA),
**IvI** Informatics Institute (UvA),
**ILLC** Institute for Logic, Language and Computation (UvA),
**NWO** Nederlandse Organisatie voor Wetenschappelijk Onderzoek,
**NTU** Nederlandse Taalunie,
**SIKS** Dutch research school for Information and Knowledge Systems,
**STEVIN** Spraak- en Taaltechnologische Essentiële Voorzieningen In het Nederlands and
**Textkernel** Textkernel b.v.

Finally, special thanks go to the following people for providing information about organizational and sponsoring matters: Alice Dijkstra, Tanja Gaustad van Zaanen, Elisabeth d'Halleweyn, Veronique Hoste, Mark Kas, Hans Kruithof, Jan Odijk, Mariet Theune, and Lisanne Teunissen.

Khalil Sima'an
Language and Computation
Institute for Logic, Language and Computation
Universiteit van Amsterdam
http://staff.science.uva.nl/∼simaan

# Invited Talk: Toward Large-Scale Shallow Semantics for Higher-Quality NLP

EDUARD HOVY

*University of Southern California, USA*

## Abstract

Building on the successes of the past decade's work on statistical methods, there are signs that continued quality improvement for QA, summarization, information extraction, and possibly even machine translation require more-elaborate and possibly even (shallow) semantic representations of text meaning. But how can one define a large-scale shallow semantic representation system and contents adequate for NLP applications, and how can one create the corpus of shallow semantic representation structures that would be required to train machine learning algorithms? This talk addresses the components required (including a symbol definition ontology and a corpus of (shallow) meaning representations) and the resources and methods one needs to build them (including existing ontologies, human annotation procedures, and a verification methodology). To illustrate these aspects, several existing and recent projects and applicable resources are described, and a research programme for the near future is outlined. Should NLP be willing to face this challenge, we may in the not-too-distant future find ourselves working with a whole new order of knowledge, namely (shallow) semantics, and doing so in increasing collaboration (after a

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

40-years separation) with specialists from the Knowledge Representation and reasoning community.

# Invited Talk: One Decade of Statistical Machine Translation: 1996–2005

HERMANN NEY

*Aachen University, Germany*

## Abstract

During the last decade, the statistical approach has found widespread use in machine translation for both written and spoken language and has had a major impact on the translation accuracy. The goal of this talk is to cover the state of the art in statistical machine translation. We will re-visit the underlying principles of the statistical approach to machine translation and summarize the progress that has been made over the last decade.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

1

# Parsing Partially Bracketed Input

MARTIJN WIELING, MARK-JAN NEDERHOF AND GERTJAN VAN NOORD

*Humanities Computing*
*University of Groningen*

## Abstract

A method is proposed to convert a Context Free Grammar to a Bracket Context Free Grammar (BCFG). A BCFG is able to parse input strings which are, in part or whole, annotated with structural information (brackets). Parsing partially bracketed strings arises naturally in several cases. One interesting application is semi-automatic treebank construction. Another application is parsing of input strings which are first annotated by a NP-chunker.

Three ways of annotating an input string with structure information are introduced: identifying a complete constituent by using a pair of round brackets, identifying the start or the end of a constituent by using square brackets and identifying the type of a constituent by subscripting the brackets with the type. If an input string is annotated with structural information and is parsed with the BCFG, the number of generated parse trees can be reduced. Only parse trees are generated which comply with the indicated structure.

An important non-trivial property of the proposed transformation is that it does not generate spurious ambiguous parse trees.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands.*
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

## 1.1 Introduction

Natural language is highly ambiguous. If natural language sentences are parsed according to a given Context Free Grammar (CFG), the number of parse trees can be enormous. If some knowledge about the type and coherence of words in a sentence is available beforehand, the number of parse trees can be reduced drastically, and the parser will be faster. In this paper we present a method to parse partially bracketed input.

The method presented in this paper is useful for a number of different applications. One interesting application is semi-automatic treebank construction. Another application is parsing of input strings which are first annotated by a syntactic chunker.

In recent years much effort is devoted to the construction of treebanks: sets of naturally occurring sentences that are associated with their correct parse. Typically, such treebanks are constructed in a semi-automatic way in which the sentence is parsed by an automatic parser, and a linguist then selects, and sometimes manually adapts, the appropriate parse from the set of parses found by the parser. If a sentence is very ambiguous this process is rather cumbersome and time consuming. In our experience in the context of the construction of the Alpino and D-Coi treebanks (van der Beek, Bouma, Malouf and van Noord 2002, van Noord, Schuurman and Vandeghinste 2006), the ability to add *some* brackets (possibly with the corresponding category) is a very intuitive and effective way to reduce annotation efforts.

Below, we also introduce the possibility to annotate a sentence with an opening bracket without a corresponding closing bracket, and vice versa. This possibility is motivated by the second application: parsing input that is pre-processed by a chunker. A chunker is an efficient program which finds occurrences of some syntactic categories (typically noun phrases). If a reliable and efficient chunker is available, syntactic parsing can be faster by using that chunker in a preprocessing stage. One common implementation strategy which goes back to Ramshaw and Marcus (1995) is to use techniques originally developed for POS-tagging, and to encode the start and end of chunks in the POS-tag inventory. Such chunkers are able to detect where a chunks starts, or where a chunk ends, but the fact that the beginning and the end of a chunk are supposed to co-occur is not inherent to the technique, but is usually added as an ad-hoc filter on the output. The ability of our method to allow independent opening and closing brackets in the input implies that this ad-hoc filter is no longer needed. It remains to be investigated if this improvement has empirical benefits as well.

In the past, researchers have experimented with techniques where pairs of parentheses are used to group constituents of an input string, such that fewer parse trees are generated. In Pereira and Schabes (1992) as well as Stolcke (1995) a method is given to adapt an existing parse algorithm (inside-outside and Earley) in such a way that it works faster with input strings which are annotated with pairs of parentheses. In McNaughton (1967) and Knuth (1967) features of a parenthesis grammar are discussed where brackets are added at the start and end of every production rule, $A \rightarrow \textbf{(}\ \alpha\ \textbf{)}$. In their bracketed context free grammar, Ginsburg and Harrison (1967) add additional information by subscripting brackets with unique indexes, $A \to [_1\ a\ ]_1$ and $A \to [_2\ b\ ]_2$.

In our research, we have focused on finding an automatic procedure to convert a given CFG to a Bracket Context Free Grammar (BCFG). A BCFG can parse the same input strings as the CFG, but in addition these input strings may be annotated in part or whole with legal structural information. By providing knowledge about the structure of an input string, the number of parse trees can be reduced and a correct parse can be found earlier. A property of the proposed transformation is that it does not generate spurious ambiguous parse trees. This property is non-trivial, as shall be shown later.

In the following section we indicate how an input string can be annotated with structural information by using brackets. In the third section a recipe is given to convert a CFG to a BCFG which can parse the annotated input strings. Features of the recipe are discussed in section 4, before the conclusion is given in section 5.

## 1.2 Annotating an input string with structural information

In previous studies (e.g. McNaughton (1967) and Knuth (1967)) structural information of the input string was added by placing a pair of brackets around each constituent (a chunk) of the input string. Our method also allows partly annotated (incomplete) input strings:

```
( The cat ) ( has caught ( a mouse ) ).
```

Three chunks can be distinguished here: `The cat`, `a mouse` and `has caught a mouse`.

It is also possible that knowledge about the type of the chunk is present (for example a noun phrase or a verb phrase, NP or VP). It should be possible to store this information, since more information about the structure may reduce the number of possible parse trees. In our model we will indicate the type of a chunk by subscripting the brackets of the chunk with this type. This way differs from Ginsburg and Harrison (1967), in which each production rule contains a pair of uniquely indexed brackets ($A \to [_1 \ldots ]_1$). Another difference is that in our annotation method incomplete input strings are possible. Note that each bracket in a pair of brackets must have the same subscript:

$$\texttt{The cat } (_{VP} \texttt{ has caught } (_{NP} \texttt{ a mouse } )_{NP}\ )_{VP}.$$

Besides allowing incomplete input strings, our method also allows for inconsistent input strings. In this case the number of opening brackets does not equal the number of closing brackets. We will use square brackets to indicate the start and/or the end of a chunk individually ([ and ]). In this case information about the structure of an input string is also present - although more limited than in the other case. Note that it is possible that an opening square bracket and a closing square bracket *may* form a chunk, as is shown in the following inconsistent input string:

$$\texttt{The cat } [_{VP} \texttt{ has caught } [_{NP} \texttt{ a mouse } ].$$

In this case it is left undecided which pair of brackets form a chunk. When certainty exists about the beginning and end of the same chunk, it is better to use the round brackets to indicate all knowledge about the structure.

Three methods can be used to indicate knowledge about the structure of an input string:

- Define a complete chunk: `(` ... `)`
- Define the start and/or end of a chunk: `[` and `]`
- Define type $A$ of a chunk: $[_A$, $]_A$, $(_A$ ... $)_A$

The three methods can be combined as can be seen in the examples below:

$$\texttt{( The cat ) [}_{\text{VP}}\texttt{ has caught (}_{\text{NP}}\texttt{ a mouse )}_{\text{NP}}\texttt{.}$$

$$\texttt{[}_{\text{VP}}\texttt{ (}_{\text{NP}}\texttt{ The mouse )}_{\text{NP}}\texttt{ walked through [}_{\text{NP}}\texttt{ the barn ].}$$

It was mentioned earlier that each single bracket in a pair of typified brackets should have the same subscript. It is also possible to subscript only one of the brackets, after which (in a separate processing step) both brackets should be given the same subscript. For this method it is necessary to find out which round brackets form a pair. This can be realised in a straightforward way. A pair of round brackets is identified by matching an opening round bracket to the nearest closing round bracket, in such a way that the number of opening round brackets equals the number of closing round brackets between them.

In this study, we have restricted ourselves to allow only structural information for non-empty chunks. This decision will be treated in more detail in section 4.

## 1.3 Converting a CFG to a BCFG

In the previous section we indicated how structural information can be added to an input string by using brackets and subscripts. The following step is to convert the original CFG to a grammar which can also parse the round and square brackets (a BCFG). Note that if the structure symbols are in the set of terminals of the original CFG, other structure symbols should be chosen.

### 1.3.1 Ambiguity problems

A first approach to create the new grammar is to generate for each production rule in the CFG, $A \rightarrow \texttt{...}$, the following 11 production rules in the BCFG $G_f$:

$$\begin{array}{lllll}
A & \rightarrow & & \texttt{...} & \\
A & \rightarrow & & \texttt{...} & \texttt{]} \\
A & \rightarrow & & \texttt{...} & \texttt{]}_A \\
A & \rightarrow & \texttt{[} & \texttt{...} & \\
A & \rightarrow & \texttt{[} & \texttt{...} & \texttt{]} \\
A & \rightarrow & \texttt{[} & \texttt{...} & \texttt{]}_A \\
A & \rightarrow & \texttt{[}_A & \texttt{...} & \\
A & \rightarrow & \texttt{[}_A & \texttt{...} & \texttt{]} \\
A & \rightarrow & \texttt{[}_A & \texttt{...} & \texttt{]}_A \\
A & \rightarrow & \texttt{(} & \texttt{...} & \texttt{)} \\
A & \rightarrow & \texttt{(}_A & \texttt{...} & \texttt{)}_A
\end{array}$$

In this way all possible configurations of brackets are represented and no parse trees will be generated which do not comply with the indicated structure. The following example illustrates this (the start symbol is $A$):

$$\begin{array}{lll}
A & \rightarrow & B\ C \\
A & \rightarrow & C\ D \\
B & \rightarrow & \mathsf{a} \\
C & \rightarrow & \mathsf{a} \\
D & \rightarrow & \mathsf{a}
\end{array}$$

The input string aa can be parsed in two ways with this grammar:

- $A \Rightarrow B\ C \overset{*}{\Rightarrow} \mathsf{aa}$
- $A \Rightarrow C\ D \overset{*}{\Rightarrow} \mathsf{aa}$

If it is known in advance that the second a is of type $D$, this can be indicated by annotating the input string in the following way: $\mathsf{a}\ \texttt{[}_D\ \mathsf{a}$. To parse this input string, the following generated production rules of $G_f$ are relevant (the other production rules are left out for simplicity):

$$\begin{array}{llll}
A & \rightarrow & & B\ C \\
A & \rightarrow & & C\ D \\
B & \rightarrow & & \mathsf{a} \\
C & \rightarrow & & \mathsf{a} \\
D & \rightarrow & & \mathsf{a} \\
D & \rightarrow & \texttt{[}_D & \mathsf{a}
\end{array}$$

The structure symbol can only be matched in the final production rule, therefore the annotated input string can be parsed in one way only: $A \Rightarrow C\ D \overset{*}{\Rightarrow} \mathsf{a}\ \texttt{[}_D\ \mathsf{a}$.

By applying this naive conversion to generate the BCFG, it is possible that for a given annotated input string a large number of spurious ambiguous parse trees are generated, which map - when the brackets are removed - on the same parse tree according to the original grammar. This is illustrated with the following CFG:

$$\begin{array}{lll} A & \rightarrow & A\ \mathtt{a} \\ A & \rightarrow & \mathtt{a} \end{array}$$

If the input string is $\mathtt{[}_A\ \mathtt{aa}$, the following generated production rules of $G_f$ are relevant:

$$\begin{array}{llll} A & \rightarrow & & A\ \mathtt{a} \\ A & \rightarrow & \mathtt{[}_A & A\ \mathtt{a} \\ A & \rightarrow & & \mathtt{a} \\ A & \rightarrow & \mathtt{[}_A & \mathtt{a} \end{array}$$

Figure 1 shows that the input string $\mathtt{[}_A\ \mathtt{aa}$ can be parsed in two ways with the BCFG, while only one parse tree exists for the unannotated input string in the original grammar. According to $G_f$ more parse trees are generated than according to the original CFG, which is of course an undesired property.

The general problem is that $G_f$ does not fix in which production rule the square bracket ([ or ]) is matched. This problem can occur with typified brackets when a production rule of the same type as the bracket is traversed multiple times before the terminal is reached. For example: $A \Rightarrow B \Rightarrow C \Rightarrow A \Rightarrow \mathtt{t}$ or $A \Rightarrow B \Rightarrow C\ \alpha \Rightarrow A\ \alpha\ \beta \Rightarrow \mathtt{t}\ \alpha\ \beta$. If the type of the (opening) bracket equals $A$, the bracket can be matched at the first or at the final production rule and multiple spurious ambiguous parse trees are generated. If the brackets are not typified this problem occurs when multiple production rules (non-terminals) are traversed before the terminal is reached. For example: $A \Rightarrow B \Rightarrow C \Rightarrow \mathtt{t}$ or $A \Rightarrow \alpha\ B \Rightarrow \alpha\ \beta\ C \Rightarrow \alpha\ \beta\ \mathtt{t}$. Because the (closing) bracket can be matched at every non-terminal, again multiple spurious ambiguous parse trees are generated.

If round brackets are used, the ambiguity problem occurs when unit rules are traversed. If the grammar is converted to Chomsky Normal Form, the problem with regard to the round brackets is solved, however the problem with the square brackets still remains.

### 1.3.2 Matching brackets as soon as possible

The problem of the previous approach, was the existence of ambiguity in the moment of matching the brackets. A solution for this problem is to define exactly when a bracket should be matched. In the following we will give a conversion of a CFG to a BCFG which enforces that brackets will be matched as soon as possible.

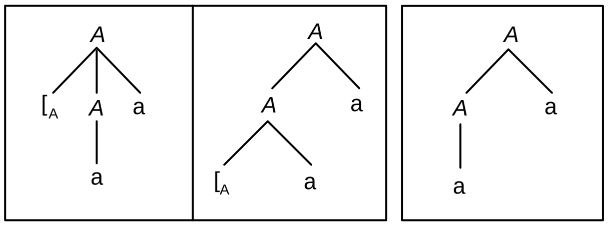


FIGURE 1 *Parse trees for input* $[_A$ aa *and* aa *(left: BCFG, right: CFG)*

**Short introduction to the method**

In the following method a large number of new production rules in the BCFG ($G$) are generated for each production rule in the CFG, based on the possible structure symbols. By using two variables ($s_0$ and $s_0'$) for each production rule in $G$, the structure symbol expected at the start ($s_0$) and at the end ($s_0'$) of the current input string are stored. Because it is not always possible to match a certain structure symbol in a production rule, it is necessary to store for each non-terminal in the right-hand side of the production the structure symbols with which these may start and end. This is done by assigning to each non-terminal in the right-hand side of the production rule two variables, which therefore map to the left side of the generated production rules. By using these variables it is enforced that if a matchable square bracket is not matched in a production rule, it can also not be matched in a later stage in the same parse tree. If round brackets are not matched, they can not be matched in a later stage as long as unit rules are encountered. A more in-depth explanation will be given after the conversion scheme is introduced.

When a specific bracket is expected as a start or end symbol of the current input string, this is indicated by setting the value of the variable ($s_0$ or $s_0'$) equal to this bracket. If no structure symbol may be matched, the symbol $\varepsilon$ is used to indicate this.

### 1.3.3 Conversion scheme CFG $\rightarrow$ BCFG

The following definitions are used with the conversion:

- $N$: the set of all non-terminals in the CFG
- $T$: the set of all terminals in the CFG
- $\Omega_{[} = \{[_A\colon A \in (N \cup \varepsilon)\}$
- $\Omega_{]} = \{]_A\colon A \in (N \cup \varepsilon)\}$
- $\Omega_{(} = \{(_A\colon A \in (N \cup \varepsilon)\}$
- $\Omega_{)} = \{)_A\colon A \in (N \cup \varepsilon)\}$
- $\Omega_b = \Omega_{[} \cup \Omega_{(} \cup \varepsilon$
- $\Omega_{b'} = \Omega_{]} \cup \Omega_{)} \cup \varepsilon$

Note that $T$ must be different from the introduced structure symbols. If this is not the case, different structure symbols must be used.

In the BCFG, we add for each production rule of the CFG

$$A \rightarrow X_1 \dots X_m$$

with $X_i \in \{N \cup T\}$, new production rules

$$A(s_0, s'_0) \rightarrow Y\ X_1{}' \dots X_m{}'\ Y'$$

with

- $(s_0, s_0{}') \in \Omega_b \times \Omega_{b'}$
- $X_i \in T \Rightarrow X_i{}' = X_i$
- $X_i \in N \Rightarrow X_i{}' = X_i(s_i, s_i{}')$, $(s_i, s_i{}') \in \Omega_b \times \Omega_{b'}$
- $Y \in \{(, [, [_A, (_A, \varepsilon\}$
- $Y' \in \{), ], ]_A, )_A, \varepsilon\}$

Where exactly one condition of 1. and one condition of 2. must hold.

For instance, to make sure an opening square bracket is matched at the first possibility, condition 1a. is used. Condition 1a. indicates that when an opening square bracket has no type or a type corresponding to the current production rule, it must be matched because $Y$ is also equal to this bracket (see condition 2a. for the closing square bracket case). Alternatively, if no structure symbol may be matched at the start of a sub-string, condition 1h. is used. Condition 1h. indicates that when no structure symbol may be matched at the start of a certain sub-string ($s_0$ equals $\varepsilon$), this will hold because $Y$ and $s_1$ must also equal $\varepsilon$ (see condition 2h. for the same case at the end of a sub-string). A detailed explanation of all conditions is given in paragraph 3.4.

1. (a) $s_0 = [_t \wedge t \in \{A, \varepsilon\} \wedge Y = s_0$
   (b) $s_0 \in \Omega_[ \setminus \{[_A, [\} \wedge X_1 \in N \wedge Y = \varepsilon \wedge s_1 = s_0$
   (c) $s_0 = (_t \wedge s_0{}' = )_t \wedge t \in \{A, \varepsilon\} \wedge Y = s_0 \wedge Y' = s_0{}'$
   (d) $s_0 = (_t \wedge s_0{}' = )_t \wedge t \in \{A, \varepsilon\} \wedge X_1 \in N \wedge m > 1 \wedge Y = \varepsilon \wedge s_1 = s_0$
   (e) $s_0 \in \Omega_( \setminus \{(_A, (\} \wedge X_1 \in N \wedge Y = \varepsilon \wedge s_1 = s_0$
   (f) $s_0 = (_t \wedge s_0{}' \neq )_t \wedge t \in \{A, \varepsilon\} \wedge X_1 \in N \wedge Y = \varepsilon \wedge s_1 = s_0$
   (g) $s_0 = \varepsilon \wedge X_1 \in T \wedge Y = \varepsilon$
   (h) $s_0 = \varepsilon \wedge X_1 \in N \wedge Y = \varepsilon \wedge s_1 = \varepsilon$
2. (a) $s_0{}' = ]_t \wedge t \in \{A, \varepsilon\} \wedge Y' = s'_0$
   (b) $s_0{}' \in \Omega_] \setminus \{]_A, ]\} \wedge X_m \in N \wedge Y' = \varepsilon \wedge s_m{}' = s_0{}'$
   (c) $s_0{}' = )_t \wedge s_0 = (_t \wedge t \in \{A, \varepsilon\} \wedge Y' = s_0{}' \wedge Y = s_0$
   (d) $s_0{}' = )_t \wedge s_0 = (_t \wedge t \in \{A, \varepsilon\} \wedge X_m \in N \wedge m > 1 \wedge Y' = \varepsilon \wedge s_m{}' = s_0{}'$
   (e) $s_0{}' \in \Omega_) \setminus \{)_A, )\} \wedge X_m \in N \wedge Y' = \varepsilon \wedge s_m{}' = s_0{}'$
   (f) $s_0{}' = )_t \wedge s_0 \neq (_t \wedge t \in \{A, \varepsilon\} \wedge X_m \in N \wedge Y' = \varepsilon \wedge s_m{}' = s_0{}'$
   (g) $s_0{}' = \varepsilon \wedge X_m \in T \wedge Y' = \varepsilon$
   (h) $s_0{}' = \varepsilon \wedge X_m \in N \wedge Y' = \varepsilon \wedge s_m{}' = \varepsilon$

An $\varepsilon$-production rule ($A \rightarrow \varepsilon$) in the CFG is converted to $A(\varepsilon, \varepsilon) \rightarrow \varepsilon$ in the BCFG. As mentioned earlier, we only allow structural information for non-empty chunks.

### 1.3.4 Explanation of the conversion scheme

For each production rule $A$ of the CFG a number of new production rules are generated in the BCFG $G$ (because of $(s_0, s_0') \in \Omega_b \times \Omega_{b'}$). For example, for a non-$\varepsilon$-production rule of a CFG:

$$A \quad \rightarrow \quad \texttt{...}$$

the conversion to $G$ will generate at least 11 new production rules:

$$\begin{array}{lllll}
A(\varepsilon\,,\,\varepsilon) & \rightarrow & & \texttt{...} & \\
A(\varepsilon\,,\,\texttt{]}) & \rightarrow & & \texttt{...} & \texttt{]} \\
A(\varepsilon\,,\,\texttt{]}_A) & \rightarrow & & \texttt{...} & \texttt{]}_A \\
A(\texttt{[}\,,\,\varepsilon) & \rightarrow & \texttt{[} & \texttt{...} & \\
A(\texttt{[}\,,\,\texttt{]}) & \rightarrow & \texttt{[} & \texttt{...} & \texttt{]} \\
A(\texttt{[}\,,\,\texttt{]}_A) & \rightarrow & \texttt{[} & \texttt{...} & \texttt{]}_A \\
A(\texttt{[}_A\,,\,\varepsilon) & \rightarrow & \texttt{[}_A & \texttt{...} & \\
A(\texttt{[}_A\,,\,\texttt{]}) & \rightarrow & \texttt{[}_A & \texttt{...} & \texttt{]} \\
A(\texttt{[}_A\,,\,\texttt{]}_A) & \rightarrow & \texttt{[}_A & \texttt{...} & \texttt{]}_A \\
A(\texttt{(}\,,\,\texttt{)}) & \rightarrow & \texttt{(} & \texttt{...} & \texttt{)} \\
A(\texttt{(}_A\,,\,\texttt{)}_A) & \rightarrow & \texttt{(}_A & \texttt{...} & \texttt{)}_A
\end{array}$$

Because non-terminals may exist in the right-hand side of the production rule $A$, it is possible that there are more production rules generated. This is discussed later.

The large number of generated production rules is necessary, because there must exist a production rule for each structure symbol in which it can be matched. If more non-terminals are present in the CFG, the number of structure symbols also increases (and this results in a larger grammar). An analysis of the number of generated production rules, based on the original production rules, the number of terminals and non-terminals in the CFG is given in a later section.

The conversion scheme enforces that terminals and non-terminals ($X_i$) remain in the same place in the generated production rule $A(s_0, s_0')$ as in the original production rule $A$.

The variables $s_0$ and $s_0'$ indicate which structure symbols are expected at the start and the end of the current input string. The variables $Y$ and $Y'$ indicate which structure symbols must be matched at the start and at the end of the current production rule.

As discussed earlier, ambiguity with respect to matching the brackets can occur with square brackets and round brackets in combination with unit-rules. This ambiguity is prevented by matching the structure symbols as soon as this is possible. The values of $Y$ and $Y'$ will therefore correspond when this is possible with $s_0$ and $s_0'$.

In the next two paragraphs the influence of $s_0$ and $s_0{}'$ on $Y$ and $s_1$ will be discussed. The situation for $Y'$ and $s_m{}'$ is analogous, instead of the conditions of 1. the conditions of 2. will be used. The relevant conditions of the conversion scheme are mentioned at the end of each paragraph.

### The influence of $s_0$ and $s_0{}'$ on $Y$

When a square bracket without a type is expected ($s_0 = [$), this symbol can be matched in every production rule and thus the value of $Y$ must equal $s_0$. This is also the case if a typified square bracket is expected with a type corresponding with the current production rule, $s_0 = [_A$ (**1a**).

When a pair of round brackets without a type is expected ($s_0 = ($ and $s_0{}' = )$), these structure symbols can be matched in every production rule. If the production rule is a unit-rule or starts and/or ends with a terminal, the values of $Y$ and $Y'$ must equal the values of $s_0$ and $s_0{}'$. If this is not the case, the values of $Y$ and $Y'$ must equal the values of $s_0$ and $s_0{}'$ or must both be equal to $\varepsilon$. Since the values of $s_0$ and $s_0{}'$ do not have to apply to the same chunk and can be matched later, the values of $Y$ and $Y'$ can also be equal to $\varepsilon$. The same arguments can be applied for a situation in which a pair of typified round brackets is expected with a type corresponding to the current production rule, $s_0 = (_A$ and $s_0{}' = )_A$ (**1c,d**).

If no structure symbol can be matched, $s_0 = \varepsilon$, $Y$ is left out (**1g,h**).

Finally, if a typified bracket is expected with a type not corresponding to the current production rule, it is not possible to match this structure symbol in the current production rule. This is also the case if a matchable opening round bracket is expected without the matchable closing round bracket. If the right-hand side of the production rule does not start with a terminal, the value of $Y$ must equal $\varepsilon$ (**1b,e,f**). In the other case no production rule is generated, because the typified bracket cannot be matched.

### The influence of $s_0$ and $s_0{}'$ on $s_1$

If no structure symbol can be matched, the current input string $w$ may not start with a structure symbol. If the right-hand side of the current production rule $A$ starts with a non-terminal $B$, the start of $w$ is parsed with the production rule belonging to $B$. Since $w$ may not start with a structure symbol, the production rule of $B$ may not start with a structure symbol. Therefore the value of $s_1$ must be equal to $\varepsilon$ (**1h**).

If a typified bracket of a different type than $A$ is expected at the start of $w$, this structure symbol cannot be matched in the current production rule. This is also the case if a matchable opening round bracket is expected without a matchable closing round bracket. In these cases the value of $s_0$, like in the previous situation, must be passed on to $B$ ($s_1 = s_0$) where the structure symbol can possibly be matched (**1b,e,f**).

If a pair of round brackets can be matched and the right-hand side of the non-unit production rule starts and ends with a non-terminal, it is also possible to pass on the round brackets. In that situation $s_1$ must be equal to $s_0$. This has to be possible, because $s_0$ and $s_0'$ can apply to different chunks and therefore should be matched later. Only in the previous three situations, the value of $s_1$ is specified. If a structure symbol is matched in the current production rule, the value of $s_1$ is free (**1a,c**).

The value of the variable $s_1'$ is free if the right-hand side of the production rule does not consist of one element (being a non-terminal). The values of the other variables $s_i$ and $s_i'$ for $1 < i < m$ are always free.

**Free variables**

If the value of one or more variables ($s_i$ and $s_i'$) is free, this results in the generation of multiple production rules for the same $A(s_0, s_0')$. For every possible combination of variable values $s_i$ and $s_i'$ a production rule must exist. This is illustrated by the following production rule (the complete CFG consists of two non-terminals):

$$A \quad \rightarrow \quad B\ \mathsf{b}$$

We limit ourselves to the generated production rules for $A([_B, \varepsilon)$. This means that at the beginning a typified bracket is expected unequal to the current type ($B \neq A$) and at the end no structure symbol may be present:

$$\begin{array}{lcl}
A([_B,\varepsilon) & \rightarrow & B([_B,\varepsilon)\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,))\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,)_A)\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,)_B)\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,])\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,]_A)\ \mathsf{b} \\
A([_B,\varepsilon) & \rightarrow & B([_B,]_B)\ \mathsf{b}
\end{array}$$

If there are more free variables present ($s_i$ or $s_i'$), this results in a significant increase of the number of production rules of $G$. This will be explained in more detail later.

Several examples of parsing an annotated input string by a BCFG are given in appendix A (downloadable from: http://www.martijnwieling.nl).

### 1.3.5 Converting generated parse trees

After the annotated input string has been parsed according to the BCFG, the final step is to convert the BCFG parse trees to CFG parse trees. This can be realized very easily by applying the following two steps (this is also illustrated in figure 2):

- Every $A(s_i, s_i')$ is replaced by $A$
- All structure symbols ($Y \neq \varepsilon$) are removed

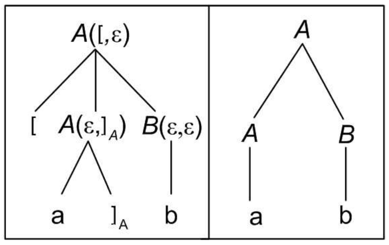


FIGURE 2 *Conversion of generated parse trees (left: BCFG, right: CFG)*

### 1.3.6 Properties of the BCFG

In this paragraph we show that the BCFG can parse all legally annotated input strings. A legally annotated input string means that there exists a parse tree in the CFG for the unannotated input string, which adheres to the structure indicated by the annotation. We also show that no extra ambiguity is caused by the annotation of the input string with structural information.

The conversion scheme enforces that terminal and non-terminal symbols remain in the same order as in the CFG. The only difference between the CFG and the BCFG is therefore the use of structural information. We therefore will focus on this aspect in the following.

- *Property 1*: The BCFG can parse all input strings which can be constructed with the CFG with the addition of legal structural information

*Proof*: The conversion scheme stores (by using $s_0$ and $s_0'$) the structure symbols with which the current input string may start and end. Because of $(s_0, s_0') \in \Omega_b \times \Omega_{b'}$ all combinations of matching structure symbols are present for every production rule and the current input string may therefore start and end with all possible structure symbol combinations. Because of $(s_i, s_i') \in \Omega_b \times \Omega_{b'}$, the non-terminals (parsing sub-strings) on the right-hand side of every production rule may also start and end with all possible structure symbol combinations. The only exception is that $s_1$ and $s_m'$ may depend on $s_0$ and $s_0'$ respectively, but this is only the case when they indicate structure symbols which are expected at the start or end of the current input string (and for this case all possible explanations were possible).

Since a square bracket or a pair of round brackets can be matched only if it does not have a type, or has a type corresponding with the current production rule (see condition a and c), only input strings can be parsed which have a legal annotation.

- *Property 2*: The BCFG does not generate CFG-equivalent parse trees for an input string.

*Proof*: CFG-equivalence of two BCFG parse trees means that if both BCFG parse trees are converted to CFG parse trees (see the previous paragraph) these parse trees are identical.

Assume there exist two BCFG parse trees for a certain annotated input string which are CFG-equivalent. In that case, it is necessary that a structure symbol is present in different places in the parse tree. This means that it must be possible to ignore a structure symbol when it can be matched first and subsequently match it in a later stage (without parsing terminals in between).

To ignore a matchable opening square bracket, the corresponding variable ($s_0$) must be equal to $\varepsilon$ (see condition g and h). This results in $s_1$, if it is present, being equal to $\varepsilon$. As a consequence, the value $s_0$ of the production rule $X_1$ will also be equal to $\varepsilon$. This process will repeat itself. To parse the input string correctly, a terminal must be matched (see condition g). This shows that it is not possible to ignore a matchable opening square bracket and match it in a later stage, before matching a terminal.

The case for a matchable closing square bracket is identical, with $s_0$ replaced by $s_0{}'$, $s_1$ by $s_m{}'$ and $X_1$ by $X_m$.

The same arguments (for $s_0$ **and** $s_0{}'$) hold for a pair of round brackets if the right-hand side of the production rule consists of one non-terminal. If this is not the case (condition d) round brackets can be ignored, but can never be matched again defining the same chunk. No production rules are generated where a single round bracket can be matched.

As we have shown, it is not possible to ignore a matchable structure symbol and match it in a later stage without matching a terminal in between. This contradicts our assumption and we can conclude that there are no CFG-equivalent parse trees generated for a certain annotated input string.

### 1.3.7 Number of generated production rules

The BCFG will consist of a large number of production rules which depends on the number of non-terminals ($N$) in the CFG, the number of non-terminals ($Z$) in the right-hand side of every single production rule and the type of $X_1$ and $X_m$ (terminal or non-terminal). An $\varepsilon$-production rule in the CFG will only generate a single production rule in the BCFG.

Four other cases can be distinguished:

1. $X_1$ and $X_m$ are both terminals
2. $X_1$ is a terminal and $X_m$ is a non-terminal, or vice versa
3. $X_1$ and $X_m$ are both non-terminals and $m > 1$
4. $X_1$ is a non-terminal and $m = 1$

When a production rule only consists of terminals, 11 production rules will be generated in the BCFG. In this case no ambiguity problem exists and the same production rules are generated as for $G_f$ (section 3). When $Z$ non-terminals are present in the production rule (not at the start and the end), $2Z$ free variables are present ($s_i$ and $s_i{}'$). Every free variable has $2N + 3$ possible values ($|\Omega_b|$ or $|\Omega_{b'}|$). The number of generated production rules in the BCFG for a production rule in the CFG which starts and ends with a non-terminal is therefore given by the following formula:

$$C_0 = 11 \cdot (2N+3)^{2Z} \tag{1.1}$$

The total number of generated production rules in the BCFG for a production rule of the CFG beginning with a terminal and ending with a non-terminal (or vice versa) is given by the following formula[1]:

$$C_1 = (22N+27) \cdot (2N+3)^{(2Z-1)} \tag{1.2}$$

For a production rule of the CFG which starts and ends with a non-terminal and $m > 1$, the following formula is used to calculate the number of generated production rules in the BCFG[1]:

$$C_2 = (44N^2 + 108N + 67) \cdot (2N+3)^{(2Z-2)} \tag{1.3}$$

When a production rule of the CFG consists only of one non-terminal ($m = 1$), the number of production rules in the BCFG is given by the following formula[1]:

$$C_3 = 44N^2 + 108N + 65 \tag{1.4}$$

For $C_0$, $C_1$, $C_2$ and $C_3$ it is clear that the number of generated production rules equals $O(N^{2Z})$. The total number of generated production rules in the BCFG based on a CFG consisting of

- $p$ production rules where $X_1$ and $X_m$ are both terminals
- $q$ production rules where $X_1$ is a terminal and $X_m$ is a non-terminal (or vice versa)
- $r$ production rules where $X_1$ and $X_m$ are both non-terminals and $m > 1$
- $s$ production rules where $X_1$ is a non-terminal and $m = 1$
- $t$ $\varepsilon$-production rules

thus equals:

$$|G| = p \cdot C_0 + q \cdot C_1 + r \cdot C_2 + s \cdot C_3 + t$$

If the original CFG is converted to Chomsky Normal Form, the right-hand side of every production rule in the CFG consists of one terminal or two non-terminals. In this case $q$, $s$ and $t$ equal 0, the value of $Z$ equals 0 for $C_0$ and the value of $Z$ equals 2 for $C_2$. The total number of generated production rules $G_c$ then equals:

$$|G_c| = p \cdot C_0 + r \cdot C_2$$

with $C_0 = 11$ and $C_2 = (44N^2 + 108N + 67) \cdot (2N+3)^2$. The number of generated production rules in $G_c$ thus has a polynomial degree, $O(N^4)$.

If the CFG is not in Chomsky Normal Form, but the highest number of non-terminals in the right-hand side of a production rule of the CFG is known ($Z_{max}$), the number of generated production rules also has a polynomial degree, $O(N^{2Z_{max}})$.

[1]A precise calculation is given in appendix B (downloadable from: http://www.martijnwieling.nl)

## 1.4 Discussion

We did not investigate in what way the size of the BCFG influences the time needed to parse an input string. Both the Earley-algorithm and the CYK-algorithm have a time complexity depending on the size of the grammar and therefore will be influenced. However, it is likely that new production rules can be generated on the fly and thus will alleviate the problem.

When it is undesirable to use a BCFG with a large number of production rules, it is also possible to use the ambiguous conversion scheme. After the parse trees have been generated according to this BCFG ($G_f$ with the addition that $\varepsilon$-production rules remain the same and do not get any structure symbols), the parse trees have to be converted to CFG parse trees by removing the structure symbols. In a subsequent sweep duplicate parse trees can be then be removed.

In our study we only allow structural information for non-empty chunks. If structural information is also desired for empty chunks, the conversion scheme cannot be adapted very easily. This is illustrated with the following example. In the production rule $A(s_0, s_0') \rightarrow X_1(s_1, s_1')\ X_2(s_2, s_2')$ the value of $X_1$ equals $\varepsilon$. A square bracket without a type can be matched in the production rule of $X_1$ (if $s_0 = [$), but it is also possible to match the square bracket in the production rule $X_2$ while not matching it in $X_1$ ($s_1 = \varepsilon$). Since the value of $s_0$ does not influence the value of $s_2$, spurious ambiguity can occur here.

## 1.5 Conclusion

We showed how an input string can be annotated with structural information and subsequently can be parsed with a BCFG. A conversion scheme was given to convert a CFG to a BCFG with the important property that the resulting BCFG does not generate spurious ambiguous parse trees.

If an input string is parsed with a CFG a large number of parse trees can be generated. The number of parse trees can be reduced by annotating the input string with structural information, parsing the annotated input string with the converted CFG (the BCFG) and converting the resulting BCFG parse trees to CFG parse trees. The number of parse trees is only reduced when the CFG contains parse trees for the original input string which do not comply to the indicated structure (these parse trees will not be generated by the BCFG).

## References

Ginsburg, S. and Harrison, M. A.(1967), Bracketed context-free languages., *J. Comput. Syst. Sci.* **1**(1), 1–23.

Knuth, D. E.(1967), A characterization of parenthesis languages, *Information and Control* **11**(3), 269–289.

McNaughton, R.(1967), Parenthesis grammars, *Journal of the ACM* **14**(3), 490–500.

Pereira, F. and Schabes, Y.(1992), Inside-outside reestimation from partially bracketed corpora, *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 128–135.

Ramshaw, L. and Marcus, M.(1995), Text chunking using transformation-based learning, *in* D. Yarovsky and K. Church (eds), *Proceedings of the Third Workshop on Very Large Corpora*, Association for Computational Linguistics, Somerset, New Jersey, pp. 82–94.

Stolcke, A.(1995), An efficient probabilistic context-free parsing algorithm that computes prefix probabilities, *Comput. Linguist.* **21**(2), 165–201.

van der Beek, L., Bouma, G., Malouf, R. and van Noord, G.(2002), The Alpino dependency treebank, *Computational Linguistics in the Netherlands*.

van Noord, G., Schuurman, I. and Vandeghinste, V.(2006), Syntactic annotation of large corpora in STEVIN, *LREC 2006*, Genua.

2

# Semantic Clustering in Dutch

**Automatically inducing semantic classes from large-scale corpora**

TIM VAN DE CRUYS

*CLCG, University of Groningen*

## Abstract

Handcrafting semantic classes is a difficult and time-consuming job, and depends on human interpretation. Unsupervised machine learning techniques might be much faster, and they do not rely on interpretation, because they stick to the data. The goal of this research is to present some clustering techniques that make it possible to automatically achieve Dutch word classes. More particularly, vector space measures are used to compute the semantic similarity of nouns according to the adjectives those nouns collocate with. Such semantic similarity measures provide a thorough basis to cluster nouns into semantic classes. Partitional clustering algorithms, that produce stand-alone clusters, as well as agglomerative clustering algorithms, that produce hierarchical trees, are investigated. For the evaluation of the clusters, evaluation frameworks will be used that compare the clusters to the hand-crafted Dutch EuroWordNet and the Interlingual Wordnet synsets. Additionally, the clustering of adjectives according to the collocating nouns has been investigated.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.
Copyright © 2007 by the individual authors.

## 2.1 Introduction

Automatically acquiring semantics from text is a subject that has gathered a lot of attention for quite some time now. As Manning and Schütze (2000) point out, most work on acquiring semantic properties of words has focused on *semantic similarity*. ‘Automatically acquiring a relative measure of how similar a word is to known words (...) is much easier than determining what the actual meaning is.’ (Manning and Schütze 2000, 295)

Most work on semantic similarity relies on the Distributional Hypothesis (Harris 1985). This hypothesis states that words that occur in similar contexts tend to be similar. Take for example the invented word *sneup*, used in a number of contexts:

- verse ‘fresh’ sneup
- gezouten ‘salty’ sneup
- lekkere ‘tasty’ sneup
- zoete ‘sweet’ sneup
- taaie ‘tough’ sneup

A speaker of Dutch who is not familiar with the word *sneup* can easily infer from the context that it is some kind of food. In the same way, a computer might be able to extract similar words from similar contexts, and group them into clusters. There are, however, some problems with such an automatic approach. Ambiguity is the most important problem. Take the examples:

(1) een oneven nummer
a odd number
‘an odd number’

(2) een steengoed nummer
a great number
‘a great song’

The word *nummer* does not have the same meaning in these examples. In example 1, *nummer* is used in the sense of ‘designator of quantity’. In example 2, it is used in the sense of ‘musical performance’. Accordingly, we would like the word *nummer* to end up in two different clusters, the first cluster consisting of words like *getal* ‘number’, *cijfer* ‘digit’ and the second cluster containing words like *liedje* ‘song’, *song* ‘song’.

While it is relatively easy for a human language user to distinguish between the two senses, this is a difficult task for a computer. Moreover, the results get blurred because the attributes of both senses (in this example *oneven* and *steengoed*) are grouped together. In 2.2.3, an approach is touched upon that might be able to resolve this kind of ambiguity. But in this research, an active disambiguation of words has not been pursued.

## 2.2 Concepts and Methodology

### 2.2.1 Vector Space Measures

The actual semantic similarity of words is determined by means of vector space measures. The two words, for which the semantic similarity is to be calculated, are represented as vectors in a multi-dimensional space. With regard to quantitative data, there are two possible vector space representations: binary vector spaces and real-valued vector spaces. **Binary vectors** only have one bit of information on each dimension, to indicate presence or absence of a feature. For linguistic objects, the **real-valued vector space** is more appropriate, as this makes it possible to encode the frequency of the attribute.

In this research, the vector space consists of the adjectives (modifiers) of the nouns. Figure 3 gives an example of four nouns represented as vectors in *modifier space*.

| | rood | lekker | snel | tweedehands |
|---|---|---|---|---|
| appel | 2 | 1 | 0 | 0 |
| wijn | 2 | 2 | 0 | 0 |
| auto | 1 | 0 | 1 | 2 |
| vrachtwagen | 1 | 0 | 1 | 1 |

FIGURE 3 A noun-by-adjective matrix

The matrix shows that the modifier *rood* collocates with all four nouns, while *lekker* only collocates with *appel* and *wijn*. On the other hand, *snel* and *tweedehands* only collocate with *auto* and *vrachtwagen*.

This example shows how it might be possible to make a judgement about the similarity of nouns according to the collocating adjectives. However, in order to make this approach really useful, an appropriate similarity measure is needed. Such a measure is discussed below.

### 2.2.2 Similarity measure

Several similarity measures are available to calculate the similarity among various patterns. A few possibilities are *Dice coefficient*, *Jaccard coefficient* and *Overlap coefficient*. An overview of various similarity measures for lexical distributional similarity is given in Weeds, Weir and McCarthy (2004). van der Plas and Bouma (2005) provide an evaluation of distributional similarity measures applied to Dutch syntactic relations.

In these experiments, the *cosine measure* has been used. The cosine measure penalizes less in cases where the number of non-zero entries is very different. This seems appropriate in the context of distributional similarity, since the amount of data available for certain words might be quite different, and we do not want to qualify words as dissimilar because of this property.

For the general case of two n-dimensional vectors $\overrightarrow{x}$ and $\overrightarrow{y}$ in a real-valued space, the cosine measure can be calculated as follows:

$$cos(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} \cdot \overrightarrow{y}}{\mid \overrightarrow{x} \mid\mid \overrightarrow{y} \mid} = \frac{\sum_{i=1}{}^{n} x_i y_i}{\sqrt{\sum_{i=1}{}^{n} {x_i}^2 \sum_{i=1}{}^{n} {y_i}^2}}$$

This formula yields a number between 0 and 1, where 0 means no similarity at all and 1 means identical vectors. When applying the cosine similarity measure to the vectors in figure 3, we get:

- $cos(appel, wijn) = \frac{6}{\sqrt{40}} \cong 0.94$
- $cos(auto, vrachtwagen) = \frac{4}{\sqrt{18}} \cong 0.94$
- $cos(appel, vrachtwagen) = \frac{2}{\sqrt{15}} \cong 0.51$

These simple examples show how semantically similar words are found: semantically similar words get high cosine values due to equal contexts of collocating adjectives, while semantically dissimilar words get a lower cosine value because of differing collocating adjectives.

### 2.2.3 Clustering

There are various clustering methods. An extensive overview of clustering is given in Jain, Murty and Flynn (1999). In general, a distinction can be made between:

- partitional clustering algorithms: algorithms that produce ‘stand-alone’ clusters which are not embedded in a structure;
- agglomerative (hierarchical) clustering algorithms: algorithms that assign a complete branching structure to the various clusters, up to the root node.

Both algorithms are worth exploring in the framework of semantic clustering. Partitional clustering is interesting to check whether similar words get grouped together. Hierarchical clustering is relevant for testing whether this kind of clustering is able to produce a sensible wordnet, that is comparable to hand-crafted wordnets. Both approaches have been explored in this paper.

#### Partitional clustering

As a partitional algorithm, K-means (MacQueen 1967) has been used. The procedure of K-means is as follows:

1. Choose *k* cluster centers, which are usually *k* randomly-chosen patterns or *k* randomly defined points inside the vector space;
2. assign each pattern to the closest cluster center;
3. recompute the cluster centers using the current cluster memberships;

4. if a convergence criterion is not met, go to step 2. Otherwise, stop the algorithm. The convergence criterium might be: no (or minimal) reassignment of patterns to new cluster centers, or a minimal decrease in squared error (a measure to check whether there are still many differences in cluster assignment in each iterative step).

### Hierarchical clustering

There are three well-known algorithms for hierarchical clustering: single-link, complete-link and group-average agglomerative clustering. These algorithms only differ in the way they characterize the similarity between a pair of clusters. In the single-link method (Sneath and Sokal 1973), the distance between two clusters is the minimum of the distances between all pairs of patterns drawn from the two clusters (one pattern from the first cluster, the other from the second). In the complete-link algorithm (King 1967), the distance between two clusters is the maximum of all pairwise distances between patterns in the two clusters. In group-average agglomerative clustering (Han and Kamber 2001), the distance between two clusters is the average distance between patterns in the two clusters. In the three algorithms, two clusters are merged to form a larger cluster based on minimum distance criteria.

This research has opted for the group-average agglomerative clustering algorithm. Group-average agglomerative clustering stands midway between single-link, which quickly merges clusters together, and complete-link, which tends to be conservative in cluster merging. The procedure of this algorithm is elaborated below:

1. The algorithm starts by taking each individual pattern in the pattern set to form a cluster;
2. next, the two clusters which are most similar are grouped together. Most similar means: the two clusters with the smallest distance between the averages of the clusters;
3. step two is repeated until there is only one cluster left. When the algorithm terminates, all clusters are hierarchically connected to the root node.

### Hard and soft clustering

An extra distinction needs to be made between hard clustering and soft clustering algorithms. In hard clustering algorithms, each element is assigned to exactly one cluster. In soft clustering algorithms, an element may be assigned to several clusters. Usually, soft clustering algorithms yield a probability distribution for each pattern, in which some patterns are more likely to belong to certain clusters than to others. This might seem a tempting approach for natural language processing, because ambiguity requires some words to be assigned to several clusters. However, soft clustering, as it is generally understood, is not the most appropriate approach to cope with disambiguous words. Since all the attributes of ambiguous words are taken into account (attributes that belong to different senses of the word), the vector that is constructed cannot represent both senses of the word, but it will present some kind of average, in which the most dominant sense will have the upper hand. Therefore, only hard clustering approaches have been pursued (equally running the risk of wrong cluster assignment with ambiguous words, and missing out on less frequent senses of an ambiguous word).

There is, however, another soft clustering approach. In this approach, an ambiguous word is first assigned to a (dominant) sense of the word (found with all the attributes of the word). Once assigned to a certain cluster, the attributes that belong to this cluster are removed from the word vector, so that other, less common senses of the word might be revealed. Such algorithms are called *disjunctive clustering models*. It might be interesting to develop such an algorithm for Dutch. The algorithm discussed by Pantel and Lin (2002) would be a good algorithm to start from. This algorithm indeed tries to find less common word senses by stripping the values of more common senses off the feature vector. However, finding less common senses of a word most likely requires much more data.

### 2.2.4 Experimental Design

All adjective-noun collocations have been extracted from the Twente Nieuws Corpus (TwNC). The corpus was tagged with Mbt (Daelemans, Zavrel et al. 1996), a memory-based tagger, and lemmatized with Mblem (van den Bosch and Daelemans 1999), a memory-based lemmatizer. The frequency of the adjectives has been logarithmically smoothed ($f(x) = 1 + ln(x) \quad for\ each\ x > 0$), in order to normalize the occurrence of many instances of one single adjective.

Various parameters have been used for clustering. A combination of the 5,000 most frequent nouns (adjectives) together with the 20,000 most frequent adjectives (nouns) was experimentally found to be functioning best (clustering more nouns yields much worse cluster quality, clustering the same nouns with more adjectives does not yield any significant improvement). This boils down to a frequency cut-off of 200 individual noun-adjective collocations per noun, while each adjective occurs at least 5 times.

## 2.3 Results

### 2.3.1 Partitional clustering

#### Noun Clustering

Below are some of the clusters that have been found by the K-means algorithm, clustering the 5,000 nouns into 700 clusters[2]:

- april januari november februari oktober maart mei juni augustus december september juli
- sprinter schaatser coureur speelster middenvelder vedette wielrenner aan-

[2]Note that the corpus has been lemmatized, but due to errors of the lemmatizer, tokens might sometimes end up in the clusters.

voerder keeper landgenoot speler atlete aanvaller renner verdediger atleet kopman bokser voetballer zwemmer spits tennisser doelman

- dollar ton euro meter kilometer kilo pond gulden centimeter
- hoofdredacteur chef commandant secretaris-generaal bestuursvoorzitter bevelhebber hoofdofficier directeur
- maand zomer winter winters week eeuw herfst
- stijger stijgers daler kanshebber winnaars boosdoener verliezer troef verliezers
- bisschop priesters Kerk predikant priester kerk dominee gelovigen kerken bisschoppen

**Adjective Clustering**

The clustering has also been done the other way around: the 5,000 most frequent adjectives have been clustered according to the 20,000 most frequent collocating nouns. Such kind of clustering produces results of the kind below:

- geel paars zwart groen blauw grijs oranje bruin roze wit rood
- Duits Amerikaans Zweeds Russisch buitenlands Brits Belgisch Nederlands Frans Engels Japans Zwitsers Italiaans Spaans Chinees
- rk roomskatholieke russisch-orthodoxe servisch-orthodoxe oud-katholiek anglicaans r.k. koptisch Koptisch grieks-orthodoxe
- zonovergoten herfstig winters druilerig zomers regenachtig zonnig
- deplorabel abominabel erbarmelijk penibel mensonterend mensonwaardig benard miserabel

### 2.3.2 Agglomerative Clustering

Agglomerative clustering also yields some remarkable results. What is remarkable is that the upper nodes present broad semantic categories, such as persons, objects, abstract entities, … Figure 4 shows part of an example of an agglomerative tree, grouping together nouns that designate a time entity. Note that the edges in the tree should not be interpreted as actual 'is a'-relations (as is the case in Wordnet), but rather as relations of semantic relatedness.

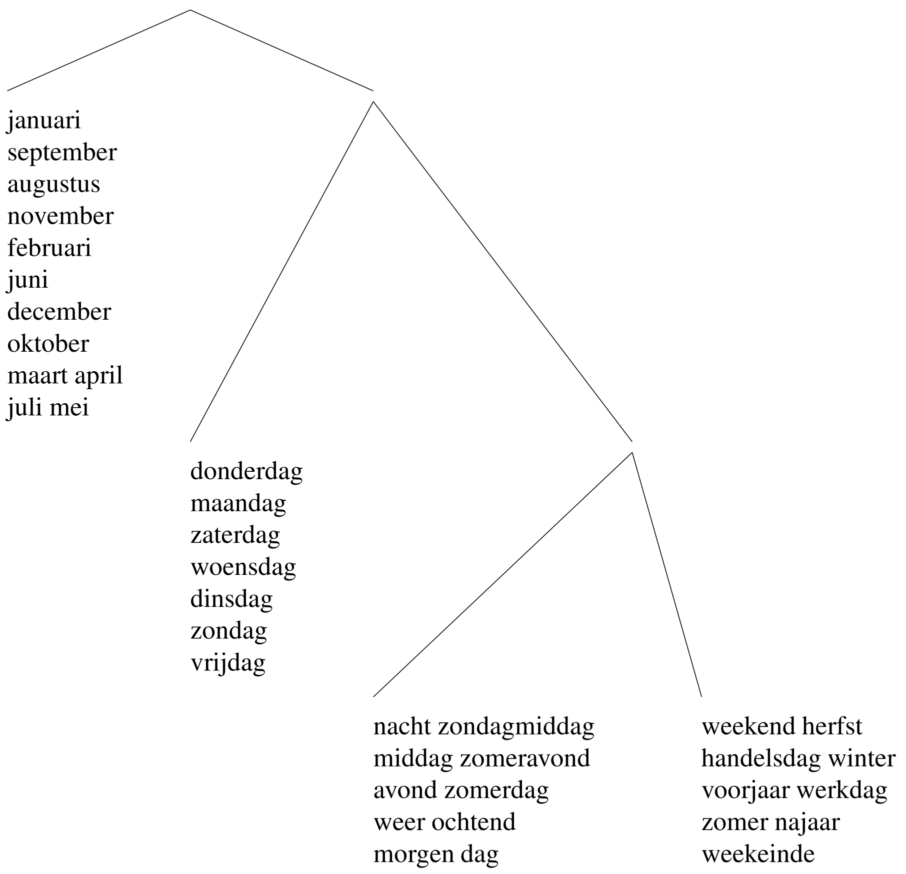


FIGURE 4 Example of agglomerative clustering: days, months, seasons

## 2.4 Evaluation

### 2.4.1 Automatic Evaluation with EuroWordNet

#### Methodology

For the evaluation of the clusters, precision and recall has been calculated according to the relations that exist in the Dutch version of EuroWordNet. The procedure of the evaluation is as follows:

- The wordnet relations that are used for the evaluation are:
  - synonyms
  - hyponyms
  - hypernyms

  - co-hyponyms (hyponyms of the hypernyms)
- For each cluster, it is checked in EuroWordNet which word from the cluster has most relations in EuroWordNet with the other words from the cluster. This word is taken to be the most central word of the cluster.
- For this word, the synonyms, hyponyms, hypernyms and co-hyponyms are drawn from EuroWordNet.
- To calculate precision, it is checked how many words from the cluster are actually appearing in the EuroWordNet-relations.
- To calculate recall, it is checked how many of the EuroWordNet-relations are not appearing in the found cluster.

A few remarks are to be made with regard to this evaluation framework. Recall will be a low number, because different kind of relations are considered in the evaluation framework: hyponyms, hypernyms, synonyms and co-hyponyms are considered all together. The real recall (as acknowledged by human judges) is probably a lot higher. To a human judge, a cluster that contains the 7 days of the week seems quite complete, but in this evaluation framework, it gets a recall of 9.21%. Therefore, recall is not such a good measure in this evaluation framework; precision will therefore be considered the most important value.

**Results**

Figure 5 presents the results of the **partitional noun clustering** evaluation. The precision and recall values are plotted against the number of clusters used.

The figures show a precision that is lower with few, large clusters, rising towards an optimal number of clusters, and then declining again when the number of clusters gets too small. Recall is faintly showing the opposite tendency, being larger with fewer but large clusters, and declining when the number of clusters get larger. But as has been explained before, recall is not such a good measure in this case.

With the optimal number of clusters, the clustering algorithm is able to reach a precision of **42.50%** (and a recall of about **8%**). Taking into account that EuroWordNet itself is incomplete (a large part of the clustered words is not known by EuroWordNet), that the evaluation algorithm is only looking one level up and down in the wordnet hierarchy, and that there is an error margin due to mistakes of the lemmatizer, the results obtained by the clustering algorithm are quite good, also given that the random baseline (results for clusters that have been randomly compiled using a hash table) is about **5.5%** for precision and about **3%** for recall.

It is interesting to have a look at the share of each relationship in the precision measure. This gives an indication of the relationships that are found by the clustering algorithm, and to what extent they are found. Figure 6 presents the share of each relationship graphically.

The majority of words found by the clustering algorithm are clearly co-hyponyms. This result was to be expected, as the horizontal relationship (which is mainly the co-hyponym

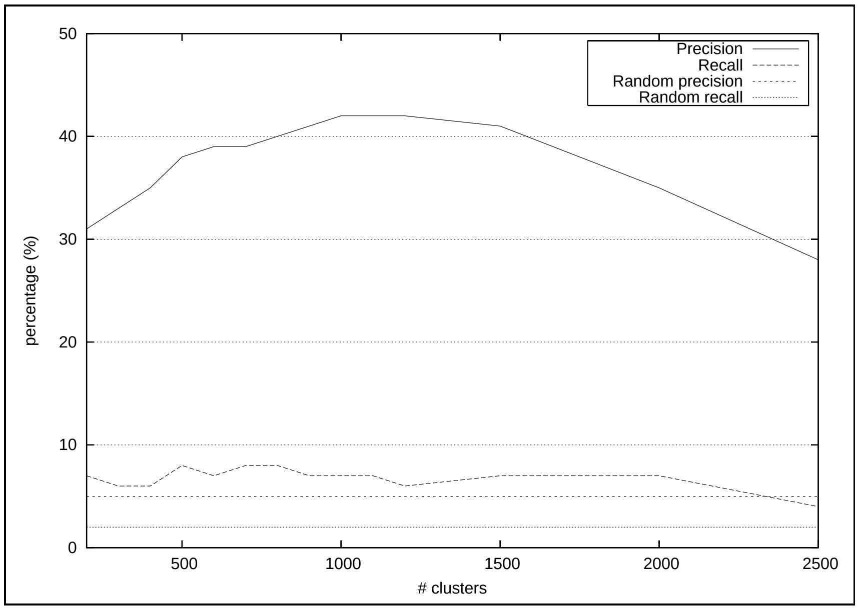


FIGURE 5 Evaluation of 5,000 nouns clustering with EuroWordNet

relationship) is the relationship in which most similarity is to be found. Synonyms (which is the second horizontal relationship) are also found by the algorithm, but the share of synonyms is of the same order as the hyponym and hypernym relationships. Most likely, this is because synonyms are less numerous than co-hyponyms. What is remarkable, is that co-hyponyms and synonyms seem to be following the same pattern: starting at a lower precision with large (fewer) clusters, rising to an optimum with middle-size clusters and declining again when the clusters get too small (too many clusters). This is not the case with the other relationships: the hypernym precision stays roughly at the same level (and is even rising a bit), while the hyponym precision is declining with the number of clusters. These results seem to indicate that hyponyms tend to get clustered more easily than hypernyms, when the margin is large enough (fewer but larger clusters). Of course, these tendencies are not significant enough to draw conclusions.

### 2.4.2 Evaluation with Wu & Palmer's measure

The algorithm discussed in 2.4.1 tries to evaluate the cluster quality by comparing the clusters to a fixed set of words extracted from EuroWordNet. This makes it possible to calculate precision and recall values. Another kind of evaluation relies on measures of **semantic similarity** according to hierarchical wordnets.

A number of such similarity measures have been developed. Among these measures, the most important are Wu & Palmer's (Wu and Palmer 1994), Resnik's (Resnik 1995) and Lin's (Lin 1998).

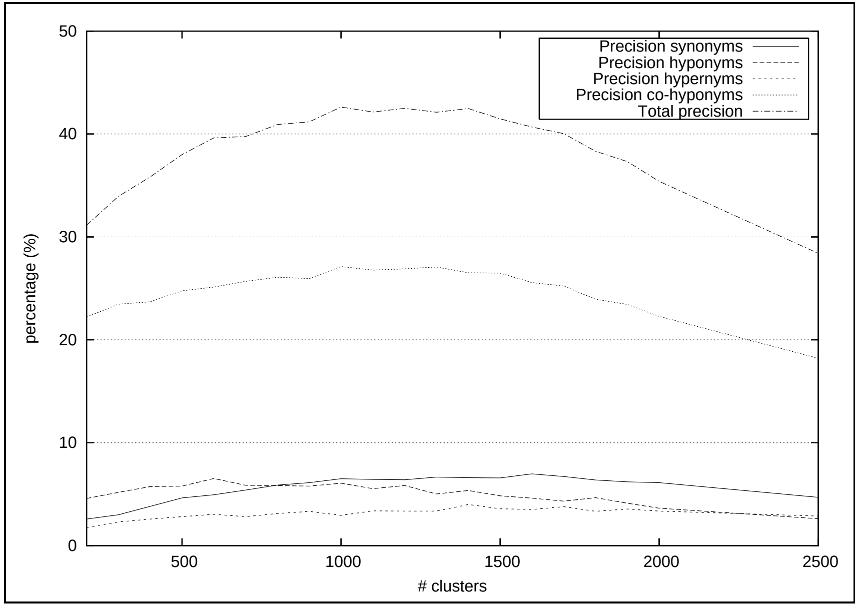


FIGURE 6 Share of each relationship in total precision

In this evaluation, Wu & Palmer's measure will be adopted. Wu and Palmer (1994) have proposed a measure that calculates the similarity between two words according to their position in a hierarchical wordnet. The similarity is calculated according to the formula given below, in which $N_1$ and $N_2$ are the number of *is-a* links from $A$ and $B$ to their most specific common superclass $C$; $N_3$ is the number of *is-a* links from $C$ to the root of the taxonomy.

$$sim_{Wu\&Palmer}(A,B) = \frac{2N_3}{N_1 + N_2 + 2N_3}$$

For example, the most common superclass of *hond* en *zalm* is *dier* (as can be seen on the extract from Dutch EuroWordNet in figure 7). Consequently, $N_1 = 2$, $N_2 = 2$, $N_3 = 4$ and $sim_{Wu\&Palmer}(hond, zalm) = 0.67$.

The results have not been calculated by using the Dutch EuroWordNet directly, as was the case with the former evaluation framework. Instead, the words have been converted to Interlingual WordNet offsets.[3] This way, it was possible to make use of a perl module that implements the computation of Wu & Palmer's measure in the English WordNet (Pedersen, Patwardhan and Michelizzi 2004).

[3]Interlingual WordNet offsets are identification codes connected with a particular WordNet synset, that have been designed to make translations among various languages possible. They are available in EuroWordNet, and are basically the same as the ones used in the English WordNet (there are differences among different versions of WordNet, but conversion procedures exist).

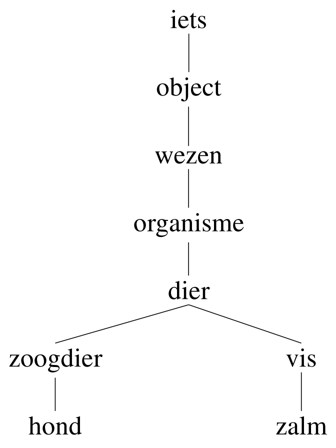


FIGURE 7 Extract from the Dutch EuroWordNet hierarchy

The average cluster similarity has been calculated as follows:

- For each cluster, the most central word has been taken (which is the word that was the closest to the cluster's centroid);
- the similarity between this most central word and every other word in the cluster has been calculated;
- If a word is ambiguous (i.e. has more than one synset), similarity has been calculated for all synsets, and the highest value has been retained;
- the average of these similarities has been taken, and every cluster average has been added up;
- all cluster averages have been divided by the total number of clusters, to get the total average;
- words not known by WordNet have been ignored.

Figure 8 shows the results of the evaluation with Wu&Palmer's measure. There's an average of 60% similarity within the clusters, while randomly compiled clusters have an average of 29% similarity. These results seem to confirm the results found by the former evaluation framework.

## 2.5 Conclusion & Further Work

In this research, clustering techniques have been explored that make it possible to automatically acquire semantic classes in Dutch. More particularly, vector space measures have been used to calculate semantic similarity. These semantic similarity measures are then used to cluster nouns into classes. Partitional K-means clustering has been used as a

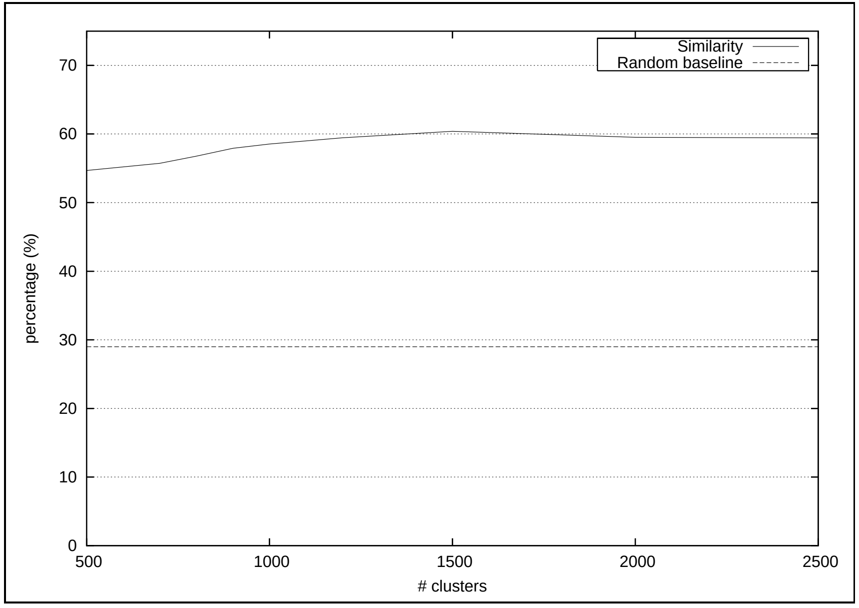


FIGURE 8 Evaluation the clustering of nouns (Wu&Palmer similarity measure)

partitional clustering algorithm and group-average agglomerative clustering has been used as a hierarchical clustering algorithm.

The results and the evaluation of the clusters have shown that using the syntactic context of words (more particularly, using modifiers to cluster nouns) is indeed a useful approach for extracting semantic classes. The evaluation of the clusters shows significant similarities with Wordnet-relations, in my own evaluation framework evaluating direct relationships, as well as with Wu & Palmer's similarity measure. The clustering of modifiers (adjectives) according to heir heads (nouns) also seems to yield quite good results, although this kind of clustering has not yet been evaluated.

There are, however, some issues that make the automatic clustering of nouns less straightforward. Ambiguity is one of the problems that is difficult to tackle for a computer. Ambiguity blurs the results, because both senses of the word (with their accompanying values) get grouped together into one sense. Disambiguating these ambiguous words into different clusters is one of the main goals to be solved, in order to reach a semantic clustering that is able to compete with hand-crafted semantic classes.

Another issue that requires more research is the automatic extraction of complete wordnets, instead of stand-alone clusters. Instead of focusing on the horizontal semantic relationships (creating clusters of similar words) it would be interesting to explore algorithms that automatically acquire the vertical relationships. This way, it might be possible to automatically construct a complete wordnet, similar to hand-crafted wordnets available. The agglomerative clustering algorithm is a first step towards this direction, though still far from perfect.

Also, the field of verb clustering remains to be explored. Subject-verb and verb-object relations are quite different from adjective-noun relations. How these relations might be used in order to cluster verbs, is subject to further research. It also remains to be investigated how these relations might help in improving the clustering of nouns.

A final interesting subject for future research is the application of dimensionality reduction techniques (LSA, PLSA) to counter data sparseness and noise. The application of these techniques will be explored in order to bring about a better clustering.

## Acknowledgements

This research has been carried out during a three month internship at the *Centrum voor Nederlandse Taal en Spraak* at the University of Antwerp. The work has been supervised by Walter Daelemans and Bart Decadt, and has substantially benefited from their remarks and contributions.

## References

Daelemans, W., Zavrel, J. et al.(1996), Mbt: A memory-based part of speech tagger-generator, *in* E. Ejerhed and I. Dagan (eds), *Proceedings of the Fourth Workshop on Very Large Corpora*, Copenhagen, Denmark, pp. 14–27.

*EuroWordNet, Building a multilingual database with wordnets for several European languages*(n.d.), website. `http://www.illc.uva.nl/EuroWordNet/`.

Han, J. and Kamber, M.(2001), *Data Mining – Concepts and Techniques*, Morgan Kaufmann, Boston.

Harris, Z.(1985), Distributional structure, *in* J. J. Katz (ed.), *The Philosophy of Linguistics*, Oxford University Press, pp. 26–47.

Jain, A. K., Murty, M. N. and Flynn, P.(1999), Data clustering: a review, *ACM Computing Surveys* **31**(3), 264–323.

King, B.(1967), Step-wise clustering procedures, *Journal of the American Statistical Association* **69**, 86–101.

Lin, D.(1998), An information-theoretic definition of similarity, *Proceedings of the 15th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, pp. 296–304.

MacQueen, J.(1967), Some methods for classification and analysis of multivariate observations, *in* L. M. Le Cam and J. Neyman (eds), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, University of California Press, Berkeley, Califonia, pp. 281–297.

Manning, C. and Schütze, H.(2000), *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Massachussets.

Pantel, P. and Lin, D.(2002), Discovering word senses from text, *Proceedings of ACM*

*Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, pp. 613–619.

Pedersen, T., Patwardhan, S. and Michelizzi, J.(2004), Wordnet::similarity - measuring the relatedness of concepts. `http://citeseer.ist.psu.edu/665035.html`.

Rennie, J.(2000), Wordnet::querydata: a Perl module for accessing the WordNet database, website. `http://people.csail.mit.edu/~jrennie/WordNet`.

Resnik, P.(1995), Using information content to evaluate semantic similarity in a taxonomy, *IJCAI*, pp. 448–453.

Sneath, P. H. A. and Sokal, R. R.(1973), *Numerical taxonomy: the principles and practice of numerical classification*, Freeman, San Francisco.

*Twente News Corpus*(n.d.), website. `http://www.vf.utwente.nl/~druid/TwNC/TwNC-main.html`.

van den Bosch, A. and Daelemans, W.(1999), Memory-based morphological analysis, *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, University of Maryland, USA, pp. 285–292.

van der Plas, L. and Bouma, G.(2005), Syntactic contexts for finding semantically similar words, *in* T. van der Wouden et al. (eds), *Computational Linguistics in the Netherlands 2004. Selected Papers from the Fifteenth CLIN Meeting*, LOT, Utrecht, pp. 173–184.

Weeds, J., Weir, D. and McCarthy, D.(2004), Characterising measures of lexical distributional similarity, *Proceedings of the 20th International Conference of Computational Linguistics*.

*WordNet, a lexical database for the English lanuguage*(n.d.), website. `http://wordnet.princeton.edu/`.

Wu, Z. and Palmer, M.(1994), Verb semantics and lexical selection, *32nd. Annual Meeting of the Association for Computational Linguistics*, New Mexico State University, Las Cruces, New Mexico, pp. 133–138.

3

# Exploring the Use of Linguistic Analysis for Answering *Why*-Questions

SUZAN VERBERNE, LOU BOVES, NELLEKE OOSTDIJK AND PETER-ARNO COPPEN

*Dept. of Linguistics, Radboud University Nijmegen*

## Abstract

In the current project, we aim at developing an approach for automatically answering *why*-questions (*why*-QA). In the present paper, we investigate the relevance of linguistic analysis for *why*-QA. We focus on two tasks: the use of syntactic information for answer type determination and the use of discourse structure for the extraction of possible answers from retrieved documents.

For answer type determination, syntactic analysis appears to be of significance: we obtain 77.5% performance using a method based on syntactic parses by the TOSCA parser—compared to 58.1% using a comparable approach without syntactic analysis.

Discourse analysis appears to be very relevant for extraction of potential answers to *why*-questions. We performed a manual analysis of 336 question-answer pairs and the corresponding RST annotated texts. We found that for 58.9% of *why*-questions, the RST analysis of the source text can lead to a correct answer to the question.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

## 3.1 Introduction

Up to now, *why*-questions have largely been ignored by researchers in the field of question answering (QA). One reason for this is that the frequency of *why*-questions in a QA context is lower than that of other types of question such as *who*- and *what*-questions (Hovy, Hermjakob and Ravichandran 2002). However, *why*-questions are not negligible: in a QA context, they comprise about 5 percent of all *wh*-questions (Hovy, Gerber, Hermjakob, Lin and Ravichandran 2001, Jijkoun and De Rijke 2005) and they do have relevance in QA applications (Maybury 2003). A second reason for disregarding *why*-questions until now is that the techniques that have proven to be successful in QA for closed-class questions are not suitable for questions that expect a procedural answer instead of a noun phrase (Kupiec 1999).

In the context of the current research into *why*-questions, a *why*-question is defined as an interrogative sentence in which the interrogative adverb *why* (or a synonymous word or phrase) occurs in (near) initial position. Furthermore, we only consider the subset of *why*-questions that could be posed to a QA system (as opposed to questions in a dialogue or in a list of frequently asked questions) and for which the answer is known to be present in some related document set.

The current paper aims to investigate the relevance of linguistic analysis for *why*-QA. Various types of linguistic analysis can be explored for analysis of both question and source text. For question analysis, we have researched the relevance of syntactic analysis for the determination of the answer type. For text analysis, we have been investigating the merits of Rhetorical Structure Theory (RST) (Mann and Thompson 1988) for extracting potential answers to *why*-questions.

In sections 2 and 3 we describe the work that we accomplished on question analysis and discourse analysis for the purpose of *why*-QA. In section 2 a syntax-based method for answer type determination is presented and evaluated. Section 3 describes the results of our study into the use of RST for the purpose of answer selection. Section 4 concludes this paper with a discussion of the plans and goals for the work that will be carried out in the remainder of the project.

## 3.2 Question analysis for *why*-QA

The goal of question analysis is to create a representation of the user's information need. The result of question analysis is an answer template that contains all information about the answer that can be induced from the question. So far, no question analysis procedures have been created for *why*-QA specifically. Therefore, we have developed an approach for the analysis of *why*-questions. In this section, we first introduce the set of *why*-questions and answers that we developed for the current research into *why*-QA. We will then present the method that we used for the analysis of *why*-questions and finally indicate the quality of our method.

### 3.2.1 Data for *why*-QA

In research in the field of QA, data sources of questions and answers play an important role. Appropriate data collections are necessary for the development and evaluation of QA systems (Voorhees and Tice 2000). In the context of the QA track of TREC, data collections in support of factoid questions have been created. However, so far, no resources have been created for *why*-QA specifically.[1] For the purpose of the present research therefore, we have developed a data collection comprising a set of questions and corresponding answers and source documents. In order to meet the requirements as formulated in Verberne et al. (2006a), it would be best to collect questions posed in an operational QA environment. Since we do not have access to such an environment, we decided to revert to the procedure used in earlier TRECs, and imitate a QA environment in an elicitation experiment.

In the elicitation experiment, ten native speakers of English were asked to read texts from ReutersŠ Textline Global News (1989) and The Guardian on CD-ROM (1992). For each text, the subjects were asked to formulate *why*-questions for which the answer can be found in the text and to formulate an answer to each of these questions. They were also asked to answer the questions of one of the other participants. The collected question-answer pairs were saved in text format, grouped per participant and per source document, so that the source information is available for each question. In this experiment, 395 questions and 769 corresponding answers were collected. For further details on the data collection, we refer to Verberne et al. (2006a).

### 3.2.2 Syntax-based analysis of *why*-questions

As described in the introduction of section 2, no approaches for the analysis of *why*-questions have been developed until now. We decided to create a syntax-based method for the analysis of *why*-questions. We will examine the relevance of syntactic analysis for question analysis by comparing our syntax-based method to an approach without the use of syntactic parsing.

In systems for factoid-QA, the answer type is generally deduced directly from the question word (*who*, *when*, *where*, etc.): *who* leads to the answer type *person*; *where* leads to the answer type *place*, etc. This information helps the system in the search for candidate answers to the question. Hovy et al. (2001) find that, of the question analysis components used by their system, the determination of the semantic answer type makes by far the largest contribution to the performance of the entire QA system.

Since determination of the semantic answer type is the most important task of existing question analysis methods, we created a question analysis method that aims to predict the answer type of *why*-questions.

In the work of Moldovan, Harabagiu, Paşca, Mihalcea, Gîrju, Goodrum and Rus (2000), all *why*-questions share the single answer type *reason*. However, we believe that

[1]There are a few sources of *why*-questions available, but these appear to be unsuitable for the aims of the present research. See for an overview of these resources Verberne, Boves, Oostdijk and Coppen (2006a).

it is useful to split this answer type into sub-types, because a more specific answer type helps the system select potential answers from the source text. The idea behind this is that every sub-type has its own lexical and syntactic cues in a source text.

Based on the classification of adverbial clauses by Quirk, Greenbaum, Leech and Svartvik (1985), we distinguish the following sub-types of *reason*:

1. *Cause* (52% of the question-answer pairs in our data collection) which is a causal relation between two events in which no deliberate human intention is involved. For example: *Why did compilers of the OED have an easier time? – Because the OED was compiled in the 19th century when language was not developing as fast as it is today.*
2. *Motivation* (37%) which adds a human intention to a causal relation. A motivation can be either a future goal or a person's internal motivation. For example: *Why has the team of researchers been split up into two teams? – To complete the work more quickly - one team will finish "A" while the second team will start on "B".*
3. *Circumstance* (2%) which adds conditionality to the temporal relation: the first event is a strict condition for the second event. For example: *Why will people buy Windows? – Because it offers more software, it is more fun to use and it works well enough.*
4. *Generic purpose* (0%) which does not express a temporal relation between two events, but gives the physical function of an object in the real world. For example: *Why do people have eyebrows? – People have eyebrows to prevent sweat running into their eyes.*

The percentages of occurrence given above are based on a manual classification of all question-answer pairs in our data collection. To the remaining 9% of question-answer pairs, we were not able to assign one of the defined answer types. A more detailed description of the answer types, the quality of the classification and their distribution in our data collection is given in Verberne et al. (2006a).

We aim at creating a question analysis module that is able to predict the expected answer type of an input question. In the analysis of factoid questions, the question word often gives necessary information about the expected answer type. In case of *why*, the question word does not give information about the answer type since all *why*-questions have *why* as the question word. This means that other information from the question is needed for determining the answer sub-type.

We decided to use Ferret's approach, in which syntactic categorization helps in determining the expected answer type. In our question analysis module, the TOSCA (TOols for Syntactic Corpus Analysis) system (Oostdijk 1996) is explored for syntactic analysis. The TOSCA (syntactic) parser takes a sequence of unambiguously POS-tagged words and assigns function and category information to all constituents in the sentence. The parser yields one or more possible output trees for (almost) all input questions. For the purpose of evaluating the maximum contribution to a classification method that can be obtained from principled syntactic analysis, we manually selected the most plausible parse tree from the parser's output. This way, we created parse trees for the 122 *why*-questions that are linked to the first three source texts in our data collection. We decided not to create parse trees for all 395 questions because creating and correcting the parse trees is quite labour-intensive. [2]

For answer type determination, we decided to use a machine learning approach exploring automatic feature selection algorithms in Weka (Holmes, Donkin and Witten 1994). These algorithms take as input a set of feature values. In our experiment, we needed a set of question features that together can predict the answer type.

As baseline we established the majority classification: a method that classifies each question in the largest class *cause*). This baseline would lead to a correct classification of 52% of the questions.

As described above, we have manually annotated our questions with their answer type. We chose five syntactic and semantic features that, based on this manual classification, seem to be relevant to the distinction between the answer types: subject agency (agentive, non-agentive), verb type (anticausative, other), modality (*can*, *have to*, *should*, etc., none), the presence and type of declarative verb (factive, semi-factive, non-factive, none), and negation (absent, present). We added four features to the feature set that give supplementary information on the syntactic structure of the question: voice (passive, active), intensive complementation construction (absent, present), monotransitive *have* construction (absent, present), and existential *there* construction (absent, present). Thus, our feature set consists of nine syntactic and semantic question features for the purpose of answer type determination.

We determined the feature values for each question by use of a Perl script that searches for patterns in the TOSCA tree. For example, the attribute *intens_compl* for the main verb of the matrix clause leads to the value 'present' for the feature *intensive complementation*. For determination of some of the features, lexical-semantic information is needed. Our script extracts this information from WordNet (Fellbaum 1998) (information on subject agency), VerbNet (Kipper, Trang Dang and Palmer 2000) (information on declaratives) and the Levin Verb Index (Levin 1993) (set of anticausatives).

The output of the script is a list of feature values for each of the 122 questions. We added the manually determined answer type for each question to complete our feature set for training. Then we used automatic feature selection algorithms to classify our questions according to their answer type.

We evaluated the classification into answer types using 10-fold cross-validation on the training set, comparing the automatically chosen answer types to the manually assigned answer types. The best-scoring algorithm (Lazy IBk) predicts 77.5% of the answer types correctly. This means our approach, classification is improved by almost 50% compared

[2] We are currently still working on evaluation of the parser. Part of this evaluation is measuring the difference in performance between automatic and manual parse selection.

to the baseline.

To check the reliability of the feature classification, we compared the outcome of the ten individual runs of the cross-validation evaluation. We found that their standard deviation to the mean is 9%. This suggests that our results are fairly reliable, despite the small data set used.

In order to investigate the merit of syntactic parsing for answer type determination, we compared the result of our syntax-based method to an approach without the use of a syntactic parser. We created a new training set consisting of feature values for the same 122 questions as in the syntax-based method. We annotated these questions with part-of-speech tags assigned by the Brill tagger. Then we used a Perl script to extract the subject, the first auxiliary and the main verb from each question—this is feasible because of the relatively uniform syntactic structure of *why*-questions. The subject, auxiliary, main verb and the question string itself serve as input for a second Perl script for determining values for the previously defined features. Again, our script uses information from WordNet, VerbNet and the Levin Verb Index to determine subject agency, verb type and declarative type. Using this input, some of the features can relatively easily be determined: subject agency, verb type, the presence and type of declaratives, and the presence of existential *there*. For intensive complementation, modality and negation, the script can make an educated guess. On the other hand, the features *voice* (passive, active) and the presence or absence of constructions with monotransitive *have* are very difficult to determine without deep parse. Still, we are confident that our script for determining the feature values has been optimized for this set of features.

We again ran the automatic feature selection algorithm to classify our questions according to their answer type, using the non-syntax-based feature set. Now the best-scoring algorithm (Naive Bayes) classifies 58.1% of questions correctly. This is an improvement of only 12% compared to the baseline. The differences between the scores for question classification with and without syntactic parsing are due to the fact that the set of feature values created with the Brill tagger contains more erroneous values. As a result, this feature set is less consistent than the set created with the TOSCA output. Due to this smaller consistency, it is more difficult for the classifier to induce rules that describe the data set.

These results show that adding syntactic parsing to an approach for determining answer types can improve its performance considerably. We therefore believe that syntactic analysis can play an important role for the analysis of *why*-questions.

### 3.3 Using RST for the purpose of *why*-QA

In section 2, we discussed the importance of answer type determination: knowing the answer type helps the QA system in selecting potential answers. After analysis of the input question, the QA system will retrieve a small set of documents that possibly contain the answer. Analysis of the retrieved documents is then needed for extracting potential answers. Thus, a system for *why*-QA needs a text analysis module that yields a set of potential answers to a given *why*-question. Although we now have a proper answer type determination approach, the problem of answer extraction is still very difficult. As opposed to factoid-QA, where named entity recognition can play an important role in extraction of potential answers, finding potential answers to *why*-questions is still an unsolved problem.

This means that we need to investigate how we can recognize the parts of a text that are potential answers to *why*-questions.

We decided to approach this answer extraction problem as a discourse analysis task. In this section, we aim to find out to what extent discourse analysis can help in selecting answers to *why*-questions. We also investigated the possibilities of a method based on textual cues, and used that approach as baseline for evaluating our discourse-based method.

We will first introduce RST as a model for discourse analysis. Then we shall present our method for investigating the use of RST for *why*-QA, followed by the results that we found. We will conclude this section with a discussion of the results, including a comparison to the baseline results, and the implications for future work.

### 3.3.1 Rhetorical Structure Theory (RST)

As framework for our research into discourse structure, we use the Rhetorical Structure Theory (RST), developed by Mann and Thompson (1988). In terms of the RST model, a rhetorical relation typically holds between two spans of text, of which one span (the *nucleus*) is more essential for the writer's intention than the other (the *satellite*). If two related spans are of equal importance, there is a *multinuclear relation* between them. Two related spans are grouped together in a larger span, which in turn can participate in a relation. The smallest units of discourse are called *elementary discourse units* (EDUs). By grouping and relating spans of text, a hierarchical structure of the document is created.

The main reasons for using RST as a model for discourse structure in the present research are the following. First, good levels of agreement have been measured between human annotators of RST, which indicates that RST is well defined (Bosma 2005). Second, a treebank of manually annotated English texts with RST structures is available for training and testing purposes. This RST Discourse Treebank, created by Carlson, Marcu and Okurowski (2003) contains a selection of 385 Wall Street Journal articles from the Penn Treebank that have been annotated with discourse structure in the framework of RST. Carlson et al. (2003) created their own set of discourse relations for use in the treebank. The annotations by Carlson et al. (2003) are largely syntax-based. They chose clauses as EDUs, using lexical and syntactic clues to help determine the clause boundaries.

### 3.3.2 Method

Let us consider a *why*-question-answer pair and the RST structure of the corresponding source text. We hypothesize the following:

1. The question topic corresponds to a span of text in the source document and the

answer corresponds to another span of text;

2. In the RST structure of the source text, an RST relation holds between the text spans representing question topic and the text span representing the answer.

If both hypotheses are true, then RST can play an important role in answering *why*-questions.

For the purpose of testing our hypotheses, we need a number of RST annotated texts and a set of question-answer pairs that are linked to these texts. Therefore, we set up an elicitation experiment using the RST treebank as data set. We followed the same elicitation method as we used for collecting data for question analysis. We selected seven texts from the RST treebank of 350–550 words each. Then we asked native speakers to read one of these texts and to formulate *why*-questions for which the answer could be found in the text. The subjects were also asked to formulate answers to each of their questions. In this experiment, they were not asked to answer one of the other participants' questions. This resulted in a set of 372 *why*-question-answer pairs, connected to seven texts from the RST treebank. On average, 53 question-answer pairs were formulated per source text. There is much overlap in the topics of the questions, as we will see later.

A risk of gathering questions following this method, is that the participants may feel forced to come up with a number of *why*-questions. This may lead to a set of questions that is not completely representative for a user's real information need. However, we believe that our elicitation method is the only way in which we can collect questions connected to a specific (closed) set of documents.

We performed a manual analysis on 336 of the collected question-answer pairs in order to check our hypotheses – we left out the other pairs for future testing purposes. We chose an approach in which we analyzed our data according to a clear step-by-step procedure, which we expect to be suitable for answer extraction performed by a future QA system. This means that our manual analysis will give us an indication of the upper bound of the performance that can be achieved using RST.

First, we selected a number of relation types from Carlson et al.'s relation set, of which we believed that they might be relevant for *why*-QA. We started with the four answer types mentioned in section 2.2, but it soon appeared that the level of detail in the relation set made it necessary to also include relations similar to cause, purpose, motivation and circumstance. Therefore, we extended the list during the manual analysis. The final set of selected relations is shown in Table 1.

TABLE 1 Selected relation types

| | | | |
|---|---|---|---|
| Cause | Circumstance | Condition | Elaboration |
| Explanation-argumentative | Interpretation | List | Problem-Solution |
| Purpose | Reason | Result | Sequence |

Then, we used the following procedure for analyzing the questions and answers:

I. Identify the topic of the question. The topic of a *why*-question is the proposition that is questioned. A *why*-question has the form 'WHY P', in which the proposition P is the topic.
II. In the RST tree of the source document, identify the span(s) of text that express(es) the same proposition as the question topic.
III. Is the found span the nucleus of a relation of one of the types listed in Table 1? If it is, go to IV. If it is not, go to V.
IV. Select the satellite of the found nucleus as answer.
V. Discard the current text span.

The effects of the procedure can best be demonstrated by means of an example. Consider the following question, formulated by one of the native speakers after he had read a text about the launch of a new TV channel: *Why does Christopher Whittle think that Channel One will have no difficulties in reaching its target?* The topic of this question is *Christopher Wittle thinks that Channel One will have no difficulties in reaching its target.* According to our first hypothesis, the proposition expressed by the question topic matches a span in the RST structure of the source document. We manually selected the following text fragment which expresses the proposition of the question topic: *What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be, said Mr. Whittle.* This sentence covers span 18–22 in the corresponding RST tree, which is shown in Figure 1 below.

In this way, we tried to identify a span of text corresponding to the question topic for each of the 336 questions. In section 3.3 we will present the results of this topic span selection step.

In cases where we succeeded in selecting a span of text in the RST tree corresponding to the question topic, we searched for potential answers following step III and IV from the analysis procedure. As we can see in Figure 1, the span *What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be, said Mr. Whittle* is the nucleus of an evidence relation. Since we assumed that an evidence relation may lead to a potential answer, we can select the satellite of this relation, span 23–28, as an answer: *He said his sales force is signing up schools at the rate of 25 a day. In California and New York, state officials have opposed Channel One. Mr. Whittle said private and parochial schools in both states will be canvassed to see if they are interested in getting the programs.*

We analyzed all 336 *why*-questions following this procedure.

### 3.3.3 Results of the analyses

As described in section 3.1, our manual analysis procedure consists of four steps: (I) identification of the question topic, (II) matching the question topic to a span of text, (III) checking whether this span is the nucleus of an RST relation, and (IV) selecting its satellite as answer. Below, we will discuss the outcome of each of these sub-tasks.

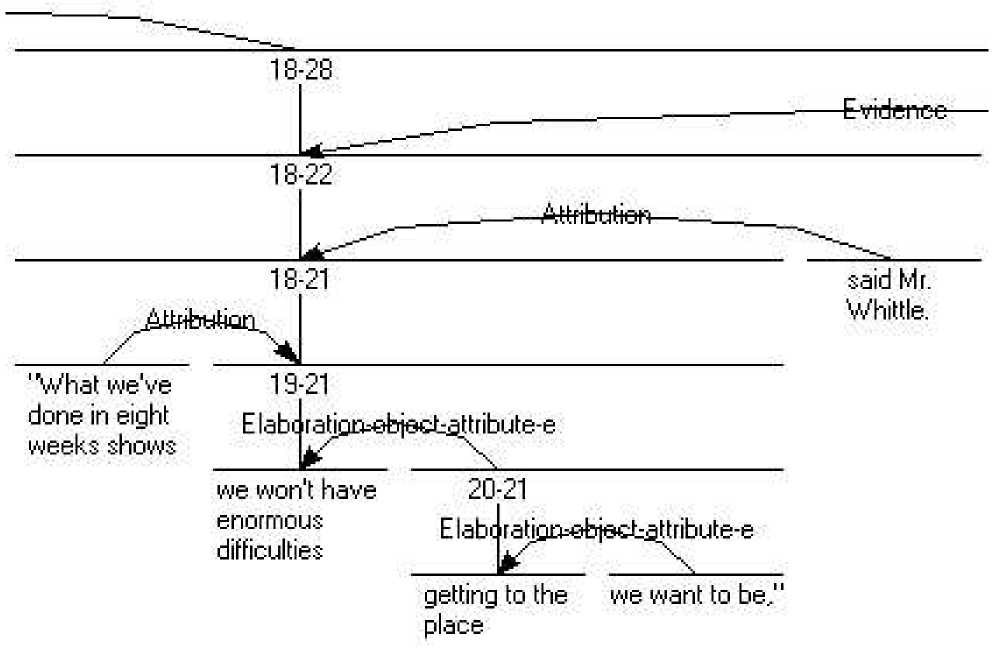

18-28
Evidence
18-22
Attribution
18-21
said Mr. Whittle.
Attribution
"What we've done in eight weeks shows
19-21
Elaboration-object-attribute-e
we won't have enormous difficulties
20-21
Elaboration-object-attribute-e
getting to the place
we want to be."

FIGURE 9 RST sub-tree for the text span "What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be, said Mr. Whittle."

The first step succeeds for all questions, since each *why*-question has a topic. For the second step, we were able to identify a text span in the source document that represents the question topic for 279 of the 336 questions that we analyzed (83.0%). We found that not every question corresponds to a unique text span in the source document. For 279 questions, we identified 84 different text spans. This means that on average, each text span that represents at least one question topic is referred to by 3.3 questions. For the other 57 questions, we were not able to identify a text span in the source document that represents the topic.

For 209 of the 279 questions that have a topic in the text (62.2% of all questions), the question topic is (part of) the nucleus of a relation of one of the types in Table 1 (step III).

Evaluation of the fourth step, answer selection, needs some more explanation. For each question, we selected as answer the satellite that is connected to the nucleus corresponding to the question topic. For the purpose of evaluating the answers found using this procedure, we compared them to the user-formulated answers. If the found answer matches at least one of the answers formulated by native speakers in meaning (not necessarily in form), then we judged the found answer as correct. For example, for the question

*Why did researchers analyze the changes in concentration of two forms of oxygen?*, two native speakers gave as an answer *To compare temperatures over the last 10,000 years*, which is exactly the answer that we found following our procedure. Therefore, we judged our answer as correct, even though eight subjects gave a different answer to this question. Evaluating the answer that we found to the question *Why does Christopher Whittle think that Channel One will have no difficulties in reaching its target?* (see above) is slightly more difficult, since it is longer than any of the answers formulated by the native speakers. However, since some of the user-formulated answers are part of the found answer span, and because the answer is still relatively short, we judged the found answer as correct.

We found that for 198 questions, the satellite connected to the nucleus corresponding to the topic is a correct answer. This is 58.9% of all questions.

### 3.3.4 Discussion and implications

#### Error analysis

We reported in section 3.3 that for 198 *why*-questions (58.9% of all questions), the answer could be found after matching the question topic to the nucleus of an RST relation and selecting its satellite as answer. This means that for 138 questions (41.1%), our method did not succeed. We distinguish four categories of questions for which we could not extract a correct answer using this method (percentages are given as part of the total of 336 questions):

1. Questions whose topics are not or only implicitly supported by the source text (57 questions, 17.0%). For example, the question *Why is cyclosporine dangerous?* refers to a source text that reads *They are also encouraged by the relatively mild side effects of FK-506, compared with cyclosporine, which can cause renal failure, morbidity, nausea and other problems.* We can deduce from this text fragment that cyclosporine is dangerous, but we need knowledge of the world (*renal failure, morbidity, nausea and other problems are dangerous*) to do this.
2. Questions for which the correct (i.e. user-formulated) answer is not or only implicitly supported by the text (15 questions, 4.5%). For example, the topic of the question *Why was Gerry Hogan interviewed?* corresponds to the text span *In an interview, Mr. Hogan said.* The native speaker who formulated this question gave as answer *Because he is closer to the activity of the relevant unit than the Chair, Ted Turner, since he has the operational role as President.* The source text does read that Mr. Hogan is president and that Ted Turner is chair, but the assumption that Gerry Hogan is closer to the activity than Ted Turner had been made by the reader; not by the text.
3. Questions for which both topic and answer are supported by the source text but the RST structure does not lead to the reference answer (55 questions, 16.4%). In some cases, this is because the topic and the answer refer to the same span. For example, the question *Why were firefighters hindered?* refers to the span *Broken water lines and gas leaks hindered firefighters' efforts,* which contains both question topic and

answer. In other cases, question topic and answer are embedded in different, non-related spans, which are often remote from each other.

4. Questions for which the topic can be identified in the text and matched to the nucleus of a relevant RST relation, but the corresponding satellite is not suitable or incomplete as answer (11 questions, 3.3%). These are the questions that in table 2 make the difference between the last two rows (209-198). Some answers are unsuitable because they are too long. In other cases, the answer satellite is incomplete compared to the user-formulated answers. For example, the topic of the question *Why did Harold Smith chain his Sagos to iron stakes?* corresponds to the nucleus of a circumstance relation that has the satellite *After three Sagos were stolen from his home in Garden Grove*. Although this satellite gives a possible answer to the question, it is incomplete according to the reference answers, which all mention the goal *To protect his trees from thieves*.

Questions of category 1 above cannot be answered by a QA system using a closed document collection. If we are not able to identify the question topic in the text manually, then a retrieval system cannot either. A comparable problem holds for questions of category 2, where the topic is supported by the source text but the answer is not or only implicitly. If the system searches for an answer that cannot be identified in a text, the system will clearly not find it in that text. In the cases where the answer is implicitly supported by the source text, knowledge of the world is often needed for deducing the answer from the text, like in the examples of cyclosporine and Gerry Hogan above. Therefore, we consider the questions of types 1 and 2 as unsolvable by any QA system that uses a closed document collection. Together these categories cover 21.4% of all *why*-questions.

Questions of category 3 (16.4% of all questions) are the cases where both question topic and answer can be identified in the text, but where the RST structure does not lead to the reference answer. We can search for ways to extent our algorithm so that it can handle some of the cases mentioned. For instance, we can add functionality for managing question-answer relations on sub-EDU level. For cases where question topic and answer are embedded non-related spans, we can at the moment not propose smart solutions that will increase recall without heavily lowering the MRR. The same holds for questions of category 4 (3.3%), where RST leads to an answer that is incomplete or unsuitable.

**Comparison to baseline**

In order to judge the value of this maximum recall, we compare the figure of 58.9% to the recall that can be achieved using our baseline approach. As baseline, we chose an approach that exploits textual cues in the source text. We performed a manual search on the 393 questions from our first data collection, their answers and the corresponding source documents. For each question-answer pair, we identified the item in the text that indicates the answer. For 50% of the questions, we could identify a word or group of words that in the given context is a cue for the answer. Most of these cues, however, are very frequent words that also occur in many non-cue contexts. For example, the subordinator *that* occurs 33 times in our document collection, only 3 of which are referred to by one or more *why*-

questions. This means that only in 9% of the cases, the subordinator *that* is a *why*-cue. The only two words for which more than 50% of the occurrences are *why*-cues, are *because* (for 18 questions) and *since* (for 9 questions). Both are a *why*-cue in 100% of their occurrences. Almost half of the question-answer pairs that do not have an explicit cue in the source text, the answer is represented by the sentence that follows (69 cases) or precedes (11 cases) the sentence that represents the question.

Having this knowledge on the frequency of cues for *why*-questions, we defined the following baseline approach:

I. Identify the topic of the question.
II. In the source document, identify the clause(s) that express(es) the same proposition as the question topic.
III. Does the clause following the matched clause start with *because* or *since*? If it does, go to IV. If it does not, go to V.
IV. Select the clause following the matched clause as answer.
V. Select the sentence following the sentence containing the matched clause as answer.

A system that follows this baseline method can obtain a maximum recall of 24.4% ((18+9+69)/393). This means that an RST-based method can improve recall by almost 150% compared to a simple cue-based method (58.9% compared to 24.4%).

### 3.3.5 RST relations that play a role in *why*-QA

We counted the number of occurrences of the relation types from Table 1 for the 198 questions where the RST relation led to a correct answer. This distribution is presented in Table 2. The meaning of the column *Relative frequency* in this context will be explained below.

As shown in table 5, the relation type with most referring question-answer pairs, is the very general elaboration relation. It seems striking that *elaboration* is more frequent as relation between a *why*-question and its answer than *reason* or *cause*. However, if we look at the relative frequency of the addressed relation types, we see another pattern: in our collection of seven source texts, *elaboration* is a very frequent relation type. In the seven texts that we consider, there are 143 occurrences of an elaboration relation. Of the 143 nuclei of these occurrences, 14 were addressed by one or more *why*-questions, which gives a relative frequency of less than 1%. *Purpose*, on the other hand, has only seven occurrences in our data collection, six of which being addressed by one or more questions, which gives a relative frequency of 0.857. *Reason* and *evidence* both have only four occurrences in the collection, three of which have been addressed by one or more questions.

The table show that if we address the problem of answer selection for *why*-questions as a discourse analysis task, the range of relation types that can lead to an answer is broad and should not be implemented too rigidly.

TABLE 2 Addressed relation types

| Relation type | # referring questions | Relative frequency |
|---|---|---|
| Means | 4 | 1.000 |
| Purpose | 28 | 0.857 |
| Consequence | 37 | 0.833 |
| Evidence | 7 | 0.750 |
| Reason | 19 | 0.750 |
| Result | 19 | 0.667 |
| Explanation-argumentative | 14 | 0.571 |
| Cause | 7 | 0.500 |
| Condition | 1 | 0.333 |
| Interpretation | 6 | 0.333 |
| Circumstance | 1 | 0.143 |
| Elaboration | 50 | 0.098 |
| Sequence | 1 | 0.091 |
| List | 4 | 0.016 |
| Problem-Solution | 0 | 0.000 |

## 3.4 Overall conclusion

We have investigated the relevance of linguistic analysis for *why*-QA. We focused on two tasks: the use of syntactic information for answer type determination and the use of discourse structure for the extraction of potential answers from retrieved documents.

For answer type determination, syntactic analysis appears to be of significance: we obtain 77.5% performance using a method based on syntactic parses by the TOSCA parser—compared to 58.1% using a similar approach without syntactic analysis.

Discourse analysis appears to be very relevant for extraction of potential answers to *why*-questions. We performed a manual analysis of 336 question-answer pairs and the corresponding RST annotated texts. We found that for 58.9% of *why*-questions, the RST analysis of the source text can lead to a correct answer to the question. Of the remaining 41.1%, there is a subset of *why*-questions (21.4% of all questions) that cannot be answered by any QA system that uses a closed document collection since knowledge of the world is essential for answering these questions. Moreover, there is a further subset of *why*-questions (16.4% + 3.3%) that cannot be answered by a system that uses RST structure only.

We should note that in a future application of *why*-QA using RST, the system will not have access to a manually annotated corpus—it has to deal with automatically annotated data. We assume that automatic RST annotations will be less complete and less precise than the manual annotations are. As a result of that, performance would decline if we were to use automatically created annotations. Some work has been done on automatically annotating text with discourse structure. Promising is the done work by Soricut and Marcu

(2003) and Huong and Abeysinghe (2003).

At present, we are working on the implementation of a system for *why*-QA that uses the manually annotated RST treebank as document collection. For the results that we obtained until now, we refer to Verberne, Boves, Oostdijk and Coppen (2006b).

## References

Bosma, W.(2005), Query-based summarization using rhetorical structure theory, *in* T. van der Wouden, M. Poß, H. Reckman and C. Cremers (eds), *15th Meeting of CLIN*, LOT, Leiden, pp. 29–44. ISBN=90-76864-91-8.

Carlson, L., Marcu, D. and Okurowski, M. E.(2003), Building a discourse-tagged corpus in the framework of rhetorical structure theory, *in* J. van Kuppevelt and R. Smith (eds), *Current Directions in Discourse and Dialogue*, Kluwer Academic Publishers, pp. 85–112.

Fellbaum, C. E. (ed.)(1998), *WordNet: An Electronic Lexical Database*, Cambridge, Mass.: MIT Press.

Holmes, G., Donkin, A. and Witten, I.(1994), Weka: a machine learning workbench, *Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems* pp. 357–361.

Hovy, E., Gerber, L., Hermjakob, U., Lin, C.-J. and Ravichandran, D.(2001), Toward semantics-based answer pinpointing, *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA.

Hovy, E., Hermjakob, U. and Ravichandran, D.(2002), A question/answer typology with surface text patterns, *Proceedings of the Human Language Technology conference (HLT)*, San Diego, CA.

Huong, T. L. and Abeysinghe, G.(2003), A Study to Improve the Efficiency of a Discourse Parsing System, *Proc of CICLing-03* pp. 104–117.

Jijkoun, V. and De Rijke, M.(2005), Retrieving answers from frequently asked questions pages on the web, *Proceedings CIKM-2005*.

Kipper, K., Trang Dang, H. and Palmer, M.(2000), Class-based construction of a verb lexicon, *AAAI-2000 Seventeenth National Conference on Artificial Intelligence*, Austin, TX.

Kupiec, J.(1999), Murax: Finding and organizing answers from text search, *Natural Language Information Retrieval* pp. 311–332.

Levin, B.(1993), *English Verb Classes and Alternations - A Preliminary Investigation*, The University of Chicago Press.

Mann, W. and Thompson, S.(1988), Rhetorical structure theory: Toward a functional theory of text organization, *Text, 8 (3)* pp. 243–281.

Maybury, M. (ed.)(2003), *Toward a Question Answering Roadmap*, pp. 8–11.

Moldovan, D., Harabagiu, S., Pasça, R., Mihalcea, R., Gîrju, R., Goodrum, R. and Rus, V.(2000), The structure and performance of an open domain question answering system, *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 563–570.

Oostdijk, N.(1996), Using the tosca analysis system to analyse a software manual corpus, *Industrial Parsing of Software Manuals* pp. 179–206.

Quirk, R., Greenbaum, S., Leech, G. and Svartvik, J.(1985), *A comprehensive grammar of the English language*, London: Longman.

Soricut, R. and Marcu, D.(2003), Sentence level discourse parsing using syntactic and lexical information, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1* pp. 149–156.

Verberne, S., Boves, L., Oostdijk, N. and Coppen, P.(2006a), Data for question answering: the case of *why*, *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy*.

Verberne, S., Boves, L., Oostdijk, N. and Coppen, P.(2006b), Discourse-based answering of *why*-questions, *Discours et document: traitements automatiques.* Submitted for *Traitement automatique des langues (TAL)*.

Voorhees, E. and Tice, D.(2000), Building a question answering test collection, *Proceedings of SIGIR-2000*, pp. 200–207.

4

# The Syntax and Semantics of Relative Clause Modification

MARKUS EGG

*Rijksuniversiteit Groningen*

## Abstract

Semantic construction for DPs with relative clauses is problematic on the basis of a surface-oriented syntactic analysis if standard tenets of the syntax-semantics interface are upheld, because due to these tenets the relative clause must be part of the NP argument of the determiner. The flexibility offered in semantic underspecification formalisms can overcome these problems. It allows semantic construction for DPs with relative clauses even if determiner and NP form a constituent first, which is modified by the relative clause. This covers the modification of indefinite pronouns like *everyone* as well as Turkish DPs with relative clauses whose determiner stands between the modifier and the NP argument. This analysis extends easily to the distinction between restrictive and non-restrictive relative clauses.

## 4.1 Introduction

Semantic construction for DPs with relative clauses is a problem for surface-oriented syntactic frameworks, where constituents are *not rearranged* on some syntactic level, if standard tenets of the syntax-semantics interface are adopted:

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.
Copyright © 2007 by the individual authors.

- semantic construction proceeds by *functional application*, which does not look into the inner structure of the semantic expression it combines with
- the involved syntactic constituents are *sisters*
- semantic contributions of constituents have the *same types* across languages[4]
  - determiners: relations between properties; $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ (Barwise and Cooper 1981)
  - nouns: properties; $\langle e,t\rangle$
  - relative clauses (and other noun modifiers like adjective phrases): functions from properties to properties; $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
  - whole DPs: sets of properties; $\langle\langle e,t\rangle,t\rangle$

The first consequence of these principles is that the only way of constructing the DP semantics is by *first* combining noun and modifier semantics. The determiner semantics comes last, its application *finishes* the process of deriving the DP semantics. Consider e.g. the semantic construction for DPs like (4.5):

(4.5) every girl whom I love

The meanings of article, noun, and relative clause are given in (4.6a-c), the meaning of (4.5) as a whole is a *set of properties* (those that every girl I love has), (4.6d):

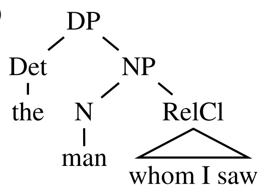
(4.6) (a) $\lambda Q \lambda P \forall x. Q(x) \rightarrow P(x)$
(b) $\lambda P \lambda x. P(x) \wedge \mathbf{love'}(\mathbf{speaker'}, x)$
(c) $\lambda x. \mathbf{girl'}(x)$
(d) $\lambda P \forall x. \mathbf{girl'}(x) \wedge \mathbf{love'}(\mathbf{speaker'}, x) \rightarrow P(x)$

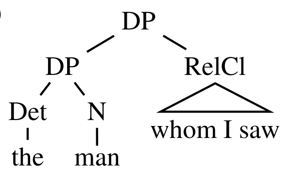
If the only way of combining the expressions (4.6a-c) into (4.6d) is functional application, then (4.6b) must be applied to (4.6c) first, the result of this application is then the argument of (4.6a). I.e., the semantics of the head noun and its modifiers must be combined first, then the resulting expression is combined with the determiner semantics, and once the semantic contribution of the determiner has been integrated, the semantics of the nominal expression cannot be augmented further.

As noted by Partee (1975), this strategy presupposes that the involved constituents form a constituent (are syntactic sisters) in the underlying (binary branching) syntactic structure. I.e., in a sequence `Det–N–RelCl`, noun and relative clause must *form a constituent first*, which is then the sister of the determiner. Following Partee, this means that the structure (4.7a) must be preferred over (4.7b):[5]

[4]For expository purposes, I will use extensional semantic representations as long as possible (i.e., up to section 4.8). A *property* is thus a set of individuals.

[5]Her tree structures have been adapted to the DP analysis of constituents such as *the man whom I love*.

(4.7) (a)


(b)


Another consequence of the tenets is that they predict a *peripheral* position of the determiner in the DP. Otherwise, the head noun could not form a constituent with the modifying relative clause. I.e., the possible orderings of determiner, nominal modifier, and head noun are as in (4.8a) and the impossible orderings as in (4.8b). E.g., English DPs exhibit the first or the second of the word orders in (4.8a):[6]

| | | | | | |
|---|---|---|---|---|---|
| (4.8) | (a) | Det Mod N | Det N Mod | Mod N Det | N Mod Det |
| | (b) | Mod Det N | N Det Mod | | |

However, the restriction to the syntactic analysis (4.7a) seems to raise problems for several kinds of DPs with relative clauses. First, Bach and Cooper (1978) and Cooper (1979) note that in DPs with *extraposed relative clauses* like in (4.9), determiner and noun must form a constituent of their own, due to the syntactic position of the relative clause. This would bar a direct combination of noun and relative clause semantics during semantic construction.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| (4.9) | Ik | heb | het | meisje | gezien | dat | je | hebt | ontmoet |
| | I | have | the | girl | seen | which | you | have | met |

‘I have seen the girl whom you have met’

*Indefinite pronouns* like *everyone* and *something* seem to comprise *both* a determiner and a noun and could thus be described as an expression of category DP. But these quantifiers can be modified by a relative clause, which contributes to the *restriction* of the quantification introduced by the modified expression, thus, (4.10) denotes the set of properties shared by every person whom the speaker loves:

(4.10) everyone whom I love

Finally, the restriction to structure (4.7a) – as well as the prediction of possible word orders – would run counter to evidence from *crosslinguistic* semantic construction, e.g., for most *Turkish* DPs with relative clauses (for similar data from other languages, see Kathol 1999). In (4.11), the nominalisation construction *sevdiğim* ‘of my loving’, which is the closest equivalent to a relative clause in English, is separated from the head noun *kız*

[6]The intransitive determiner *a* can exceptionally be preceded by APs including a specifier like *too* or *how* and *such*, but this order is very idiosyncratic as it does not generalise to other APs and other determiners in English, consider e.g. *every such function* and **good a man*.

'girl' by the determiner *her* 'every':[7]

| | | | | | |
|---|---|---|---|---|---|
| (4.11) | sev- | diğ- | -im | her | kız |
| | love | NOM | POSS.1Sg | every | girl |

'every girl whom I love' (literally, 'of my loving every girl')

In this paper, I will present an analysis of DPs with relative clauses that relinquishes the condition that semantic construction should proceed by functional application. The proposed analysis is not the first one to do so, but it is novel in that it allows semantic construction on the basis of structure (4.7b) without additional assumptions like syntactic decomposition of indefinite pronouns and Cooper Storage mechanisms (see section 4.2). What is more, the approach extends naturally to the distinction between restrictive and non-restrictive relative clauses as in (4.12) and (4.13):

(4.12) The train which leaves at 11:30am is waiting on platform 5

(4.13) The train, which leaves at 11:30am, is waiting on platform 5

While the latter sentence entails that there is only one train (which happens to leave at 11:30am), the departure time is (at least, pragmatically) necessary in (4.12) to distinguish the denoted train from other trains.

The paper is structured as follows: First I will discuss previous analyses of the problematic examples, then I will sketch my intuitions on the examples presented in this introduction and outline the general framework of the flexible syntax-semantics interface of Egg (2004). Then I will extend this interface to DPs with relative clauses (including non-restrictive relative clauses) and conclude with an outlook on further topics in the syntax-semantics interface for relative clauses.

## 4.2 Previous analyses

As soon as the techniques of semantic construction are no longer limited to functional application, (4.7a) is no longer the only syntactic structure on which a syntax-semantics interface for DPs with relative clauses can be built.

The analyses of Cooper (1979) and Janssen (1983) for the extraposed relative clauses in Hittite regard the matrix clause and the relative clause as syntactic sisters. The restriction of the semantics of the DP the relative clause belongs to contains a *predicate variable*, which is inherited by the semantics of the matrix clause. Subsequent $\lambda$-abstraction over this variable turns the semantics of the matrix clause into a function whose application to the semantic contribution of the relative clause yields the desired semantic interpretation.

(4.14) illustrates the semantic construction for (4.9) in terms of such an analysis. The semantics of main clause, relative clause, and the result of constructing the semantics for

[7] In the gloss, 'NOM' is short for 'nominalisation,' 'POSS.1Sg' indicates a possessive suffix of the first person singular, the Turkish equivalent to English *my*.

(4.9) as a whole are given in (4.14a-c). $\exists!x.[P(x)] \wedge Q(x)$ abbreviates $\exists x.(\forall y.P(y) \rightarrow y = x) \wedge Q(x)$:

(4.14) (a) $\exists!x.[\mathbf{girl'}(x) \wedge R(x)] \wedge \mathbf{see'}(\mathbf{speaker'}, x)$
(b) $\lambda y.\mathbf{meet'}(\mathbf{hearer'}, y)$
(c) $\lambda R\big(\exists!x.[\mathbf{girl'}(x) \wedge R(x)] \wedge \mathbf{see'}(\mathbf{speaker'}, x)\big)\big(\lambda y.\mathbf{meet'}(\mathbf{hearer'}, y)\big) =$

$$\exists!x.[\mathbf{girl'}(x) \wedge \mathbf{meet'}(\mathbf{hearer'}, x)] \wedge \mathbf{see'}(\mathbf{speaker'}, x)$$

Bach and Cooper (1978) and Janssen (1983) extend this analysis (introducing a predicate variable in the semantics of the modified constituent) to DPs with relative clauses whose structure is (4.7b). The variable indicates where the modifier semantics is to be integrated; the integration proceeds via $\lambda$-abstraction over the variable. But such abstractions usually occur in the semantics of a subcategorising constituent to indicate where the semantic contribution of a subcategorised constituent should go. Using them for the integration of the semantics of a *modifier* thus treats modifiers as complements just for the sake of semantic construction.

Treating modifiers as complements has been suggested before (e.g., in Bouma, Malouf and Sag 2001), but has been motivated on independent syntactic grounds (e.g., extraction data). Since I will show that an analysis of the data is possible that need not treat modifying relative clauses as complements, further syntactic evidence should be adduced to back up this treatment of modifiers.

The modification of indefinite pronouns like in (4.10) has given rise to analyses that derive the meaning of the resulting constituents from a not directly visible level of syntax. Abney (1987) suggests a movement analysis in which a (bound) noun, e.g., *one*, is first modified by the relative clause and then incorporated with the determiner after subsequent head-to-head movement.[8] Such analyses (see also Kishimoto 2000, Sag 1997) can remain faithful to the abovementioned principles and still assume an underlying syntactic structure of the type (4.7a). For advocates of surface-oriented syntactic frameworks, however, no such strategy is open.

Data like (4.11) - as well as the exceptional English DPs noted in footnote 6 - could be handled by Janssen's (1983) 'Det-S' analysis of DPs with relative clauses or in Ginzburg and Sag's (2000) account of exceptional Mod-Det ordering as in *too big a house*: Here determiner and modifier combine first, and the result is combined with the noun (phrase). The analyses show that the syntax-semantics interface can derive the desired semantic structures from this syntactic structure with rules much more complex than mere functional application.

Ginzburg and Sag (2000) attribute this structure for the English cases to a *lexical* prop-

[8]Abney uses the fact that indefinite pronouns are morphologically transparent in English. But this does not hold universally, consider e.g. German *jemand* 'someone'. This would enforce different analyses of the phenomenon in English and German, which seems unintuitive. He must also stipulate an ambiguity of words like *one* or *body* between a free and a bound variant with considerably different interpretations.

erty of the indefinite English determiner (optional selection for a gradable AP), which is justified by the idiosyncrasy of the construction in English. But then examples like (4.11), where there is no such idiosyncrasy, call for a different analysis.

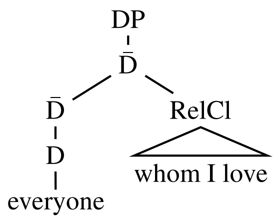
Finally, the question of how to do semantic construction for DPs with relative clauses in HPSG is not yet settled. Kathol's claim that the HPSG syntax-semantics interface can use either structure in (4.7) holds only for versions of HPSG with quantifier storage (e.g., Pollard and Sag 1994, Ginzburg and Sag 2000) where the semantic content of a determiner expression is the one of its NP argument. In more recent versions of the theory where Minimal Recursion Semantics (Copestake, Flickinger, Pollard and Sag 2005) is used as the semantic representation formalism, however, semantic construction on the basis of (4.7b) would run into problems.[9]

In the following, I will show how to do semantic construction for DPs with relative clauses on the basis of the syntactic structure (4.7b). Following Kathol (1999) and Egg (2004), I assign this syntactic structure to (4.10). This analysis extends naturally to (4.11), it could be used for further kinds of DPs with relative clauses (as argued for e.g. by Kathol 1999), but I will not discuss this question in this paper.

## 4.3 Underlying intuitions

This section will further analyse the examples presented in the introduction on an intuitive level, the formalisation will then be presented in sections 4.4-4.7. Starting point is the analysis of Egg (2004) for the semantic construction of (4.10). Indefinite pronouns emerge as intransitive determiners, and there is adjunction of the relative clause to the pronoun (after projection to the bar level), which then projects to DP:

(4.15)



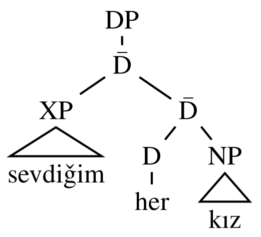
The mismatch between syntactic and semantic structure is handled in the syntax-semantics interface. The challenge is the derivation of (4.16a), the semantics of (4.10), from the pronoun meaning (4.16b) and the modifier meaning (4.16c). Applying (4.16c) to the underlined part of (4.16b) only would yield (4.16a).[10]

[9]At the level of D̄ or DP, the NP constituent would no longer be visible for semantic construction, because its HANDLE value (its 'address') is no longer available as the LTOP value.

[10]Merely type-lifting the relative clause to an expression of type $\langle\langle\langle e,t\rangle,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ would not suffice: This strategy could not by itself make sure that the modifier ends up as part of the restriction, in the scope of the quantification introduced by the pronoun.

(4.16) (a) $\lambda P \forall x.\mathbf{person'}(x) \wedge \mathbf{love'}(\mathbf{speaker}, x) \rightarrow P(x)$
(b) $\lambda P \forall x.\underline{\mathbf{person'}}(x) \rightarrow P(x)$
(c) $\lambda P \lambda x.\overline{P(x) \wedge} \mathbf{love'}(\mathbf{speaker}, x)$

The Turkish example exhibits a very similar syntactic structure: Since Turkish DPs can consist of determiner and noun alone (e.g., *her kız* ‘every girl’), it makes sense to assume that they form a D̄-constituent in (4.11), too. Then the XP *sevdiğim* (whose category is not relevant for the presented analysis) adjoins to this constituent, and, finally, it projects to DP:

(4.17)


Now the challenge is the derivation of the semantic representation of (4.11) on the basis of (4.17). However, as soon as the striking resemblance of (4.17) to the structure (4.15) of example (4.10) is taken into account (modification of a D̄ constituent in either case), it is possible to reformulate the challenge in analogy to (4.16): How can (4.18a), the semantic representation of (4.11), be derived using (4.18b) and (4.18c), the meanings of the modified D̄-constituent and of the XP, respectively?

(4.18) (a) $\lambda P \forall x.\mathbf{love'}(\mathbf{speaker}, x) \wedge \mathbf{girl'}(x) \rightarrow P(x)$
(b) $\lambda P \forall x.\underline{\mathbf{girl'}}(x) \rightarrow P(x)$
(c) $\lambda P \lambda x.\overline{\mathbf{love}}'(\mathbf{speaker'}, x) \wedge P(x)$

Semantic application of (4.18c) to (4.18b) or vice versa is not possible, however, if one could apply (4.18c) only to the underlined part of the D̄ semantics, which indicates the semantic contribution of the NP, one would immediately get (4.18a). I.e., once again we encounter the question of how to apply the semantics of a modifier only to the restriction of a quantification introduced in the modified expression.

The close similarity between the syntactic and semantic structures of (4.10) and (4.11) suggests that the syntax-semantics mapping works in a very similar way. Since the modified *her kız* in (4.11) is syntactically *complex*, it will be derived by appropriate interface rules. The output of these rules should have the same structure as the lexical entry for the semantics of an indefinite pronoun. In this way, one could handle the semantic effect of modifying Turkish Det-N constituents as well as of English indefinite pronouns by one single rule of the syntax-semantics interface.

The next topic addressed in the introduction were *non-restrictive* relative clauses. Following the intuition of Bartsch (1979), non-restrictive relative clauses are seen as part of the *scope* of the quantifier. The differences in the interpretation of restrictive and non-restrictive relative clauses follow immediately, consider the semantic representations for (4.12) and (4.13) in (4.19a) and (4.19b).

(4.19) (a) $\exists! x[\mathbf{train'}(x) \wedge \mathbf{leave\text{-}at\text{-}11{:}30am'}(x)] \wedge \mathbf{wait\text{-}on\text{-}p5'}(x)$
(b) $\exists! x[\mathbf{train'}(x)] \wedge \mathbf{wait\text{-}on\text{-}p5'}(x) \wedge \mathbf{leave\text{-}at\text{-}11{:}30am'}(x)$

The property with respect to which the denoted entity in (4.19b) is unique is being a train, i.e., there are no other trains. In contrast, the denoted entity in (4.19a) is the unique train leaving at 11:30 am, thus, there might be others.

The difference between these semantic representations is put down to the specific intonation for nonrestrictive relative clauses as indicated by the commas.[11]

## 4.4 The formalism

In this section, the *underspecification formalism* Constraint Language for Lambda Structures CLLS; (CLLS; Egg, Koller and Niehren 2001), on which the proposed analysis is built, is expounded in an abbreviated form. Its expressions are (meta-level) *constraints* that describe a set of semantic representations. In this paper, the described semantic representations are $\lambda$-terms.

Semantic representations are called *solutions* of a constraint when they are described by or compatible with it. For the present paper, we can restrict ourselves to a subset of solutions, viz., those that comprise only material explicitly mentioned in the constraint. Under this restriction, constraints can be regarded as a *partial order* on a set of fragments of semantic representations. There are many other underspecification formalisms, e.g., Lexical Resource Semantics (LRS; Richter and Sailer 2004), or Minimal Recursion Semantics (MRS; Copestake et al. 2005).

If one uses such a formalism in the syntax-semantics interface, one can derive the different readings of a structurally ambiguous expression by mapping one (surface-oriented) syntactic structure to one constraint $C$ such that the set of semantic representations described by $C$ corresponds exactly to the readings of the expression. This strategy has been successfully used for accounts of scope ambiguity. My claim is that it can be applied to DPs with relative clauses, too (even if these DPs are themselves not structurally ambiguous), because it makes possible an extremely flexible syntax-semantics interface.

E.g., the result of the syntax-semantics mapping for (4.10) is the CLLS constraint (4.20). Please ignore for the time being any labels like '$[\![\mathtt{C}]\!]$', I will explain them in section 4.5.

[11] See Carlson (1977) for further syntactic differences between the two kinds of relative clauses.

(4.20)

$$\begin{tikzcd}
 & \llbracket \mathsf{DP} \rrbracket : \boxed{\cdot} \arrow[dl, dotted, no head] \arrow[dr, dotted, no head] & \\
\llbracket \mathsf{DP_S} \rrbracket : \lambda P \forall x.\ \boxed{\cdot}(x) \rightarrow P(x) \arrow[dr, dotted, no head] & & \lambda y.\ \boxed{\cdot}(y) \wedge \mathbf{love'}(\mathbf{speaker'}, y) \arrow[dl, dotted, no head] \\
 & \mathbf{person'} &
\end{tikzcd}$$

This constraint illustrates the ingredients of simplified CLLS expressions:

- *fragments* of $\lambda$-terms
- not yet known parts of these fragments, indicated by '*holes*'
- *dominance relations* (depicted by dotted lines) that link fragments to holes[12]

Dominance relations between a fragment and a hole indicate that the fragment is an (im-)proper part of what the hole stands for. These dominance relations model scope, and are therefore also used to model quantifier scope ambiguities.

In the constraint (4.20), the semantic representation for the relative clause constitutes the right hand fragment. The meaning of the pronoun is expressed in the left and the bottom fragment, which together make up (4.16b). The underlined part of (4.16b), viz., the restriction of the quantification, emerges as a fragment of its own in (4.20). This bottom fragment is dominated by the right and the left fragment, thus, ends up in the scope of both the quantification and the modifier. The scope between the right and the left fragment is undecided, since they are both dominated by the same hole at the top. Structures like (4.20) are called *dominance diamonds*.

The fact that there is only a hole on top in (4.20) indicates that we do not yet know what a $\lambda$-term described by the constraint looks like. However, due to the dominance relations between this hole and the fragments on the right and the left we know that these fragments are the immediate parts of such a $\lambda$-term.

To resolve the ambiguity in constraints, information is added monotonically, in particular, by strengthening dominance relations between holes and fragments to *identity*. For (4.20), there is only one choice, viz., identifying the bottom fragment with the hole in the modifier fragment, the modifier fragment, with the hole in the quantifier fragment, and the quantifier fragment, with the top hole. This returns (4.21), where the relative clause pertains only to the restriction of the quantification introduced by the pronoun.

(4.21) [= (4.16a)] $\lambda P \forall x.\mathbf{person'}(x) \wedge \mathbf{love'}(\mathbf{speaker'}, x) \rightarrow P(x)$

The other option (which starts by identifying the bottom fragment with the hole in the quantifier fragment) is ruled out by the types of the involved fragments. I.e., (4.20) is an adequate representation of the semantics of (4.10) in spite of the potential ambiguity it

[12]I will talk about fragment $F_1$ dominating another fragment $F_2$ (instead of talking about dominance between a hole $h$ in $F_1$ and $F_2$) when the identity of $h$ is clear from the context.

expresses; it does not overgenerate, as unwanted ambiguity is blocked.

## 4.5 The interface rules

In this part of the paper, I will use this formalism to define a very flexible syntax-semantics interface which allows the derivation of constraints like (4.20) on the basis of a surface-oriented underlying syntactic analysis. The interface presumes that the semantic contribution of every syntactic constituent is *structured* in that it distinguishes a *main* and an *embedded* fragment. In CLLS constraints like (4.20), '$[\![\mathsf{C}]\!]$' indicates the main fragment of a constituent $C$ and '$[\![\mathsf{C_S}]\!]$', the secondary fragment of $C$. '$[\![\mathsf{C}]\!]\,\mathsf{:F}$' expresses that the main fragment of $C$ is defined as fragment $F$. Consider e.g. the lexical entry for the semantics of *everyone*, where the restriction of the quantification is singled out as the secondary fragment, while the rest of the semantic representation shows up in the main fragment:

(4.22)
$$\begin{tikzcd} {[\![\mathsf{D}]\!] : \lambda P \forall x.\ \Box\ (x) \rightarrow P(x)} \arrow[d, dash, dotted] \\ {[\![\mathsf{D_S}]\!] : \mathbf{person}'} \end{tikzcd}$$

Interface rules specify for a constituent $C$ how the constraints $Con_1$ and $Con_2$ of its immediate constituents $C_1$ and $C_2$, inherited by $C$, are combined into a new constraint for $C$. The main and the secondary fragments of $Con_1$ and $Con_2$ are accessible to the rules; they combine $Con_1$ and $Con_2$ by addressing their main and secondary fragments and subsequently determine these features for $C$. For instance, the simple rule that non-branching $\bar{\text{X}}$ constituents inherit their fragments from their heads is written as (4.23). In the syntactic part of such rules, a subscripted label after the opening bracket indicates the category of the constituent $C$:

(4.23) $$[_{\bar{\mathsf{X}}}\ \mathsf{X}] \qquad \overset{\text{(SSI)}}{\Rightarrow} \qquad [\![\bar{\mathsf{X}}]\!] : [\![\mathsf{X}]\!]; \qquad [\![\bar{\mathsf{X}}_\mathsf{S}]\!] : [\![\mathsf{X_S}]\!]$$

The semantic representation of modification (adjunction) structures like (4.10) and (4.11) is constructed by the interface rule (4.24). The main fragment $[\![\bar{\mathsf{X}}_1]\!]$ of the whole constituent is defined as $[\![\bar{\mathsf{X}}_2]\!]$, the one from the modified expression. In contrast, its secondary fragment $[\![\bar{\mathsf{X}}_{1\mathsf{S}}]\!]$ is not inherited from this expression, instead, it consists of an application of the modifier fragment $[\![\mathsf{Mod}]\!]$ to a hole that dominates the secondary fragment $[\![\bar{\mathsf{X}}_{2\mathsf{S}}]\!]$ of the modified expression. This yields the bottom half of a dominance diamond, in which $[\![\mathsf{Mod}]\!]$ and $[\![\bar{\mathsf{X}}_1]\!]$ are scopally ambiguous in that they both dominate $[\![\bar{\mathsf{X}}_{2\mathsf{S}}]\!]$. Dominance between $[\![\bar{\mathsf{X}}_1]\!]$ and $[\![\bar{\mathsf{X}}_{2\mathsf{S}}]\!]$ is specified in the semantic representation of $\bar{\text{X}}_2$ (recall that $[\![\bar{\mathsf{X}}_1]\!]$ is equated with $[\![\bar{\mathsf{X}}_2]\!]$), e.g., it can eventually follow from lexical entries as (4.22). The equation of the modifier fragments ($[\![\mathsf{Mod}]\!]$: $[\![\mathsf{Mod_S}]\!]$) is introduced to facilitate reading.

(4.24) $[_{\bar{\texttt{X}}_1}\texttt{Mod}\ \bar{\texttt{X}}_2]$ $\overset{\text{(SSI)}}{\Rightarrow}$

$$\begin{tikzcd}[column sep=small]
[\![\bar{\texttt{X}}_{1\texttt{S}}]\!] : [\![\texttt{Mod}]\!](\boxed{\cdot}) \arrow[d, dash, dotted] & [\![\texttt{Mod}]\!] : [\![\texttt{Mod}_\texttt{S}]\!]; [\![\bar{\texttt{X}}_1]\!] : [\![\bar{\texttt{X}}_2]\!] \\
[\![\bar{\texttt{X}}_{2\texttt{S}}]\!] &
\end{tikzcd}$$

The rule that constructs the upper half of the dominance diamond corresponds to the syntax rule `XP -> X̄`. The main fragment of the XP is a hole that dominates both fragments of the X̄ constituent:

(4.25) $[_{\texttt{XP}}\ \bar{\texttt{X}}]$ $\overset{\text{(SSI)}}{\Rightarrow}$

$$\begin{tikzcd}[column sep=small]
& [\![\texttt{XP}]\!] : \boxed{\cdot} \arrow[dl, dash, dotted] \arrow[dr, dash, dotted] & \\
[\![\texttt{XP}_\texttt{S}]\!] : [\![\bar{\texttt{X}}]\!] & & [\![\bar{\texttt{X}}_\texttt{S}]\!]
\end{tikzcd}$$

Semantic construction for *everyone I love* now starts with the semantic representations (4.22) and (4.26) of pronoun and relative clause (whose derivation is omitted here). Rule (4.23) states that (4.22) is the semantics of the D̄ *everyone* too.

(4.26) $[\![\texttt{RelCl}]\!], [\![\texttt{RelCl}_\texttt{S}]\!]$: $\lambda P \lambda x. P(x) \wedge \mathbf{love}'(\mathbf{speaker}', x)$

Rule (4.24) combines the constraints of D̄ and RelCl into (4.27), the lower half of a dominance diamond as a consequence of adjoining the relative clause to the D̄ *everyone*. This lower half serves as input to rule (4.25), which yields the dominance diamond (4.20). Rule (4.25) applies because the D̄ *everyone that I love* projects to DP.

(4.27)

$$\begin{tikzcd}[column sep=small]
[\![\bar{\texttt{D}}]\!] : \lambda P \forall x. \boxed{\cdot}(x) \rightarrow P(x) \arrow[dr, dash, dotted] & & [\![\bar{\texttt{D}}_\texttt{S}]\!] : \lambda x. \boxed{\cdot}(x) \wedge \mathbf{love}'(\mathbf{speaker}', x) \arrow[dl, dash, dotted] \\
& \mathbf{person}' &
\end{tikzcd}$$

## 4.6 The analysis

The interface rules (4.23)-(4.25) can now be reused for the syntax-semantics interface of Turkish. These rules handle the modification of the D̄ constituent *her kız* (and the subsequent projection of the complete modification structure to DP); in addition, we need one more rule to describe the semantic consequence of forming D̄ constituents out of a determiner and its NP complement: The main fragment of such a D̄ constituent is the main D fragment applied to the main NP fragment; the NP's secondary fragment becomes the one of the D̄.

(4.28) $[_{\bar{\texttt{D}}}\ \texttt{D}\ \texttt{NP}]$ $\overset{\text{(SSI)}}{\Rightarrow}$ $[\![\bar{\texttt{D}}]\!]$: $[\![\texttt{D}]\!]([\![\texttt{NP}]\!])$; $[\![\bar{\texttt{D}}_\texttt{S}]\!]$: $[\![\texttt{NP}_\texttt{S}]\!]$

Semantic construction for (4.11) will assign it a semantic representation in analogy to the construction of (4.10). In particular, the secondary fragment of the modified expression will be the restriction of a quantifier in its primary fragment. We start with simple lexical entries for the meaning of *her* and *kız*, respectively:

(4.29) (a) $[\![\mathsf{D}]\!]$, $[\![\mathsf{D_S}]\!]$: $\lambda Q\lambda P\forall x.Q(x) \rightarrow P(x)$
(b) $[\![\mathsf{N}]\!]$, $[\![\mathsf{N_S}]\!]$: $\mathbf{girl}'$

According to (4.23) these meanings are inherited to the projections of *her* and *kız* to $\bar{\mathrm{D}}$ and $\bar{\mathrm{N}}$ level. Next comes the derivation of (4.30), the semantics of *kız* as a NP constituent, which involves rule (4.25). Then rule (4.28) combines (4.30) with the semantics of *her* to derive (4.31), the semantics of *her kız* as $\bar{\mathrm{D}}$:

(4.30)
$$\begin{tikzcd}[\![\mathsf{NP}]\!] : \boxed{\cdot} \arrow[d, dash, dotted] \\ [\![\mathsf{NP_S}]\!] : \mathbf{girl}'\end{tikzcd}$$

(4.31)
$$\begin{tikzcd}[\![\bar{\mathsf{D}}]\!] : \lambda P\forall x.\ \boxed{\cdot}\ (x) \rightarrow P(x) \arrow[d, dash, dotted] \\ [\![\bar{\mathsf{D}}_\mathsf{S}]\!] : \mathbf{girl}'\end{tikzcd}$$

(4.31) is the desired input for the interface rule for modification. The restriction of the quantifier introduced by the $\bar{\mathrm{D}}$ emerges as the embedded fragment of this constituent, which can then be addressed by the modification interface rule (4.24).

This rule combines (4.31) and (4.32), the semantics of *sevdiğim*, into (4.33), the semantics of the $\bar{\mathrm{D}}$ expression *sevdiğim her kız.* This constraint constitutes the lower half of a dominance diamond.

(4.32) $[\![\mathsf{XP}]\!]$,$[\![\mathsf{XP_S}]\!]$: $\lambda P\lambda x.P(x) \wedge \mathbf{love}'(\mathbf{speaker}', x)$

(4.33)
$$\begin{tikzcd}[\![\bar{\mathsf{D}}]\!] : \lambda P\forall x.\boxed{\cdot}(x) \rightarrow P(x) \arrow[dr, dash, dotted] & & [\![\bar{\mathsf{D}}_\mathsf{S}]\!] : \lambda y.\boxed{\cdot}(y) \wedge \mathbf{love}'(\mathbf{speaker}', y) \arrow[dl, dash, dotted] \\ & \mathbf{girl}' & \end{tikzcd}$$

Rule (4.25) then yields the dominance diamond (4.34) on the basis of (4.33). The sole solution of this constraint is (4.35).

(4.34)
$$\begin{tikzcd} & [\![\mathsf{DP}]\!] : \boxed{\cdot} \arrow[dl, dash, dotted] \arrow[dr, dash, dotted] & \\ [\![\mathsf{DP_S}]\!] : \lambda P\forall x.\ \boxed{\cdot}\ (x) \rightarrow P(x) \arrow[dr, dash, dotted] & & \lambda y.\boxed{\cdot}(y) \wedge \mathbf{love}'(\mathbf{speaker}', y) \arrow[dl, dash, dotted] \\ & \mathbf{girl}' & \end{tikzcd}$$

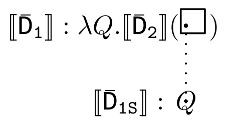
(4.35) [= (4.18a)] $\lambda P\forall x.\mathbf{girl}'(x) \wedge \mathbf{love}'(\mathbf{speaker}', x) \rightarrow P(x)$

In sum, a flexible syntax-semantics interface allows a unified semantic analysis of relative clause modification in typologically diverse languages. In the following section, this analysis will be extended to non-restrictive relative clauses.

## 4.7 Non-restrictive relative clauses

On the basis of this general approach to relative clauses we can now sketch a treatment of nonrestrictive relative clauses. To this end, we need a rule that is the semantic correlate of the specific intonation that distinguishes non-restrictive relative clauses. Since the written reflex of the intonation is a comma between the modified noun and the relative clauses, I will (roughly) characterise the rule that prepares $\bar{\mathrm{D}}$ constituents for modification by a non-restrictive relative clauses as a rule that 'interprets the comma'. I.e., the rule defines the semantics of a constituent $\bar{\mathrm{D}}_1$ consisting of a constituent $\bar{\mathrm{D}}_2$ and a comma in terms of the semantics of $\bar{\mathrm{D}}_2$:

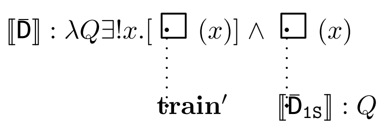
(4.36) $[_{\bar{\mathtt{D}}_1}\ \bar{\mathtt{D}}_2,]$ $\overset{\text{(SSI)}}{\Rightarrow}$



$[\![\bar{\mathtt{D}}_1]\!] : \lambda Q.[\![\bar{\mathtt{D}}_2]\!](\boxed{\cdot})$
$[\![\bar{\mathtt{D}}_{1\mathtt{S}}]\!] : Q$

In the resulting constraint, the secondary fragment is defined as the *scope* of the quantification, no longer as its restriction. Consider e.g. the result of applying (4.36) to the semantic representation of *the train*:

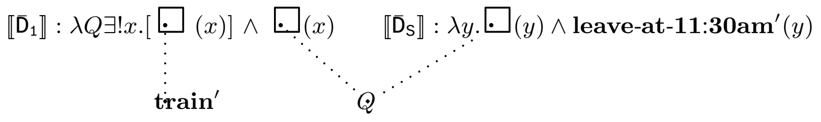
(4.37)



$[\![\bar{\mathtt{D}}]\!] : \lambda Q\exists!x.[\ \boxed{\cdot}\ (x)] \wedge\ \boxed{\cdot}\ (x)$
$\mathbf{train}'$
$[\![\bar{\mathtt{D}}_{1\mathtt{S}}]\!] : Q$

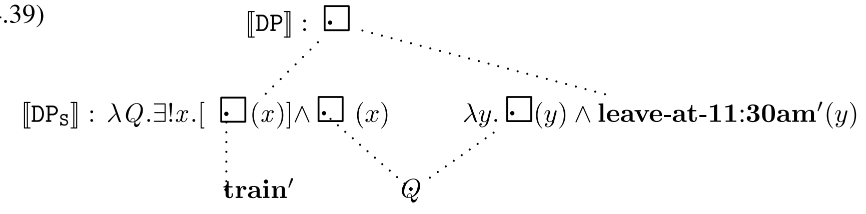
Subsequent modification, which pertains to the secondary fragment of the modified expression, will then end up in the scope of the quantifier. As an example, the result of modifying *the train* by the non-restrictive *which leaves at 11:30am* according to the rules (4.24) and (4.25) is sketched in (4.38) and (4.39):

(4.38)



$[\![\bar{\mathtt{D}}_1]\!] : \lambda Q\exists!x.[\ \boxed{\cdot}\ (x)] \wedge\ \boxed{\cdot}(x)$
$[\![\bar{\mathtt{D}}_\mathtt{S}]\!] : \lambda y.\boxed{\cdot}(y) \wedge \mathbf{leave\text{-}at\text{-}11{:}30am}'(y)$
$\mathbf{train}'$
$Q$

(4.39)



$[\![\mathtt{DP}]\!] : \boxed{\cdot}$
$[\![\mathtt{DP}_\mathtt{S}]\!] : \lambda Q.\exists!x.[\ \boxed{\cdot}(x)]\wedge\boxed{\cdot}\ (x)$
$\lambda y.\ \boxed{\cdot}(y) \wedge \mathbf{leave\text{-}at\text{-}11{:}30am}'(y)$
$\mathbf{train}'$
$Q$

The only solution for this constraint is (4.40), the set of properties such that the unique train has them in addition to the property of leaving at 11:30am:

(4.40) $\lambda Q\exists!x.[\mathbf{train}'(x)] \wedge Q(x) \wedge \mathbf{leave\text{-}at\text{-}11{:}30am}'(x)$

With the solution we have sketched so far, we can tackle a related phenomenon, viz., modification of proper names as in (4.41):

(4.41) Bill, who is a friend of mine

Once again one must combine an expression of type $\langle\langle e,t\rangle,t\rangle$ with a function from properties to properties into another expression of type $\langle\langle e,t\rangle,t\rangle$. But with the help of rule (4.36), semantic construction for (4.41) is straightforward.

Starting from the semantics of *Bill* as sketched in (4.42),[13] the application of (4.36) to (4.42) returns (4.43), where the scope of *Bill* emerges as secondary fragment.

(4.42) $[\![\mathtt{D}]\!], [\![\mathtt{D_S}]\!]: \quad \lambda P.P(\mathbf{bill}')$

(4.43)
$$\begin{tikzcd}
{[\![\bar{\mathtt{D}}_1]\!] : \lambda Q.\boxdot(\mathbf{bill}')} \arrow[d, dash, dotted] \\
{[\![\bar{\mathtt{D}}_{1S}]\!] : Q}
\end{tikzcd}$$

This constraint can then be combined with the semantics of the relative clause into a dominance diamond following the rules (4.23)-(4.25):

(4.44)
$$\begin{tikzcd}
& {[\![\mathtt{DP}]\!] : \boxdot} \arrow[dl, dash, dotted] \arrow[dr, dash, dotted] & \\
{[\![\mathtt{DP_S}]\!] : \lambda Q.\ \boxdot\ (\mathbf{bill}')} \arrow[dr, dash, dotted] & & {\lambda y.\ \boxdot(y) \wedge \mathbf{friend}'(y, \mathbf{speaker}')} \arrow[dl, dash, dotted] \\
& Q &
\end{tikzcd}$$

The sole solution of this diamond is the desired $\lambda$-term (4.45), the set of Bill's properties apart from being a friend of the speaker:

(4.45) $\lambda Q.Q(\mathbf{bill}') \wedge \mathbf{friend}'(\mathbf{bill}', \mathbf{speaker}')$

At first glance, the modification of proper nouns seems to resemble the modification of indefinite pronouns as in (4.10). If one would model the semantics of proper names by an explicit existential quantification (e.g., as the set of properties such that an entity named *Bill* has them), one could indeed handle (4.41) in analogy to (4.10), viz., by letting the relative clause pertain semantically only to the restriction of the modified expression (the property of being named *Bill*).

However, such a strategy would fail to take into account the difference between the two phenomena, viz., the fact that the modification of indefinite pronouns can involve restrictive and non-restrictive relative clauses.[14] In contrast, proper names can only be

[13]The syntactic characterisation of proper names as intransitive determiners is motivated by the fact that they can form DPs by themselves, can be modified, and are incompatible with determiners.

[14]Here are some examples of indefinite pronouns modified by non-restrictive relative clauses:

modified by non-restrictive relative clauses, because the proper name itself suffices for the identification of its referent.[15]

## 4.8 Outlook

The analysis sketched in this paper captures a number of challenging cases for the semantic construction of expressions with relative clauses, but there remain enough issues for further research. One such issue that seems to be fruitfully analysable in terms of a suitable extension of the proposed analysis is the interaction of the relative clause with a nominal expression that it modifies.

Consider one of the examples discussed by Bhatt (2002). This DP has two readings, one referring to the first book of which John claimed that Tolstoy wrote it (relevant is the order of *claiming*), and the other, to the book that - according to John - is the first book that Tolstoy wrote (relevant is the order of *writing*):

(4.46) the first book that John said that Tolstoy wrote

The two readings are due to a scope ambiguity between *first* and *say*, but the first reading raises the question of how material from the modified nominal expression *first book* can end up in the scope of an operator in the relative clause.

Bhatt (2002) assumes movement of this expression from within the embedded relative clause *that Tolstoy wrote*. The interpretation of such structures chooses one of the elements of this movement chain (movement proceeds by copying) and ignores the others. I.e., one can interpret the NP locally, in the scope of *say*.

While it is pretty straightforward to represent the two readings in a dominance diamond like (4.47), whose solutions model the two readings of (4.46), the question of how to derive the constraint (4.47) in the presented analysis is still open.

(i) Everyone in this room, who has worked together for this common goal, should celebrate (`http://www.chnonline.org/2002/2002-07-04/newsstory4.html`)

(ii) Each and everyone in this room, who are members of The Grand Lodge of Canada in the Province of Ontario (`http://freemasonry.org/nking/Orient\%20Lodge\%20-\%20Education.htm`)

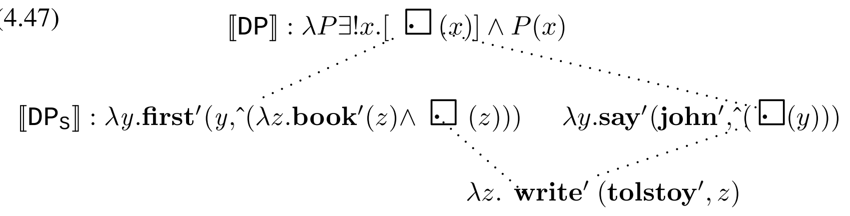
[15]One reviewer adduces sentences like (i) as counterargument to this claim:

(iii) I'm talking about Kim who works in the sales department, not about Kim who works in the cleaning team

I do share the reviewer's intuition that the relative clause has a restricting effect in (iii) in that it identifies the person talked about. However, I attribute this effect to the fact that there is constrastive focus marking on *sales department*. According e.g. to the Alternative Semantics approach to focus (Rooth 1992), this means that the first sentence in (iii) implies that all contextual alternatives to this sentence (where the speaker talks about a person named Kim, and this Kim works in another contextually relevant place) are ruled out. If there are several Kims in the context, this restriction will identify the relevant one.

In addition, some native speakers accept sentences like (iii) only with definite articles before the proper names. In this case, the counterargument does not even arise, because the use of the article shows that proper names are reanalysed as proper nouns here (for the name *X*, the set of persons named *X*).

(4.47)


$[\![\mathsf{DP}]\!] : \lambda P \exists ! x.[\ \boxed{\cdot}\ (x)] \wedge P(x)$
$[\![\mathsf{DP_S}]\!] : \lambda y.\mathbf{first'}(y, \hat{}(\lambda z.\mathbf{book'}(z) \wedge \ \boxed{\cdot}\ (z)))$
$\lambda y.\mathbf{say'}(\mathbf{john'}, \hat{}(\boxed{\cdot}(y)))$
$\lambda z.\ \mathbf{write'}\ (\mathbf{tolstoy'}, z)$

In this analysis, $\mathbf{first'}(x, P)$ is true for a world-time pair $\langle w, t\rangle$ iff $P(x)$ is true for $\langle w, t\rangle$ and for all $x' \neq x$ such that $P(x')$ for a $\langle w, t'\rangle$ it follows that $t < t'$ (in prose: $x$ is the first entity with the property $P$).

(4.48) (a) $\lambda P \exists ! x.[\mathbf{say'}(\mathbf{john'}, \hat{}\mathbf{first'}(x, \hat{}(\lambda z.\mathbf{book'}(z) \wedge \mathbf{write'}(\mathbf{tolstoy'}, z))))] \wedge P(x)$

(b) $\lambda P \exists ! x.[\mathbf{first'}(x, \hat{}(\lambda z.\mathbf{book'}(z) \wedge \mathbf{say'}(\mathbf{john'}, \hat{}\mathbf{write'}(\mathbf{tolstoy'}, z))))] \wedge P(x)$

(4.48a-b) stand for the properties of an entity that is unique with respect to a specific property. In (4.48a), it is the property of being designated by John as the first book written by Tolstoy. In (4.48a), it is the property of being the first book that John designates as being written by Tolstoy.

In (4.48a), the semantic contributions of the modified NP *first book* and the modifying relative clause are interleaved in such a way that the semantics of the modified expression is inserted within the modifier semantics, which is exactly the reverse pattern from the analysis of modification sketched in the analyses of (4.10) and (4.11). Further work is necessary to develop a suitable syntax-semantics interface to derive semantic structures like (4.47) from surface-oriented syntactic analyses.

## References

Abney, S.(1987), *The English noun phrase in its sentential aspect*, PhD thesis, MIT.

Bach, E. and Cooper, R.(1978), The NP-S analysis of relative clauses and compositional semantics, *Linguistics and Philosophy* **2**, 145–150.

Bartsch, R.(1979), The syntax and semantics of subordinate clause constructions and pronominal coreference, *in* F. Heny and H. Schnelle (eds), *Syntax and Semantics 10*, Academic Press, New York, pp. 23–59.

Barwise, J. and Cooper, R.(1981), Generalized quantifiers and natural language, *Linguistics & Philosophy* **4**, 159–219.

Bhatt, R.(2002), The raising analysis of relative clauses: evidence from adjectival modification, *Natural Language Semantics* **10**, 43–90.

Bouma, G., Malouf, R. and Sag, I.(2001), Satisfying constraints on adjunction and extraction, *Natural Language and Linguistic Theory* **19**, 1–65.

Carlson, G.(1977), Amount relatives, *Language* **53**, 520–542.

Cooper, R.(1979), The interpretation of pronouns, *in* F. Heny and H. Schnelle (eds), *Syntax and Semantics 10*, Academic Press, New York, pp. 61–93.

Copestake, A., Flickinger, D., Pollard, C. and Sag, I.(2005), Minimal Recursion Semantics. An introduction, *Research on Language and Computation* **3**, 281–332.

Egg, M.(2004), Mismatches at the syntax-semantics interface, *in* S. Müller (ed.), *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Publications, Stanford, pp. 119–139.

Egg, M., Koller, A. and Niehren, J.(2001), The Constraint Language for Lambda-Structures, *Journal of Logic, Language, and Information* **10**, 457–485.

Ginzburg, J. and Sag, I.(2000), *Interrogative investigations*, CSLI Publications, Stanford.

Janssen, T.(1983), *Foundations and applications of Montague Grammar*, PhD thesis, Mathematisch Centrum, Amsterdam.

Kathol, A.(1999), Restrictive modification and polycategorial NPs in English, *in* G. Bouma et al. (eds), *Constraints and resources in natural language syntax and semantics*, CSLI Publications, Stanford, pp. 83–100.

Kishimoto, H.(2000), Indefinite pronouns and overt N-raising, *Linguistic Inquiry* **31**, 557–566.

Partee, B.(1975), Montague Grammar and Transformational Grammar, *Linguistic Inquiry* **6**, 203–300.

Pollard, C. and Sag, I.(1994), *Head-driven Phrase Structure Grammar*, CSLI and University of Chicago Press.

Richter, F. and Sailer, M.(2004), Basic concepts of Lexical Resource Semantics, *in* A. Beckmann and N. Preining (eds), *European Summer School in Logic, Language and Information 2003*, Kurt Gödel Society, Wien, pp. 87–143.

Rooth, M.(1992), A theory of focus interpretation, *Natural Language Semantics* **1**, 75–116.

Sag, I.(1997), English relative clause constructions, *Journal of Linguistics* **33**, 431–484.

5

# The Contours of a Semantic Annotation Scheme for Dutch

INEKE SCHUURMAN AND PAOLA MONACHESI

*Centrum voor Computerlinguï stiek, K.U.Leuven*
*Maria-Theresiastraat 21, 3000 Leuven, Belgium*
*ineke.schuurman@ccl.kuleuven.be*
*and*
*Utrecht University, Uil-OTS*
*Trans 10, 3512 JK Utrecht, The Netherlands*
*Paola.Monachesi@let.uu.nl*

## Abstract

The creation of semantically annotated corpora has lagged dramatically behind. As a result, the need for such resources has now become urgent. Several initiatives have been launched at the international level in the last years, however, they have focussed almost entirely on English and not much attention has been dedicated to the creation of semantically annotated Dutch corpora. The Flemish-Dutch STEVIN-programme has identified semantic annotation as one of its priorities.Within the project "Dutch Language Corpus Initiative" (D-Coi) we are developing guidelines for the semantic annotation of Dutch and our focus is on two types: semantic role assignment and temporal and spatial semantics.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

## 5.1 Introduction

The realization of an appropriate digital language infrastructure for Dutch is one of the objectives of the Dutch/Flemish STEVIN programme which has been recently launched.[16] In particular, the need for a large corpus of written Dutch, comprising 500-million-words has been identified as one of the top priorities. This corpus should be tailored to the needs of scientific research and commercial applications and should improve the development of other resources and tools. Applications such as information extraction, question-answering, document classification, and automatic abstracting that are based on underlying probabilistic techniques should benefit from it.

All texts in the corpus will conform to standards for character encoding and markup. Furthermore, the corpus will be linguistically annotated. For the various annotation layers, annotation schemes must be decided upon and the aim is to revise and adapt the protocols which have been developed for the Spoken Dutch Corpus (CGN) (Oostdijk, Goedertier, Van Eynde, Boves, Martens, Moortgat and Baayen 2002).

A pilot study is being carried out to this end: the *Dutch language Corpus Initiative* (D-Coi) is a project launched within the STEVIN programme whose aim is a blueprint for the construction of the 500-million-word corpus.[17] The project is concerned with issues related to the design of the corpus and the development (or adaptation) of protocols, procedures and tools that are needed for sampling data, text regularization, converting file formats, marking up, annotating, post-editing, and validating the data. Within the D-Coi project, a 50 million word pilot corpus will be compiled, parts of which will be enriched with (verified) linguistic annotations. The pilot corpus is intended to demonstrate the feasibility of the approach. It will provide the necessary testing ground on the basis of which feedback can be obtained about the adequacy and practicability of various annotation schemes and procedures, and the level of success with which tools can be applied.

One of the innovative aspects of the D-Coi project is that it will focus not only on the revisions of those protocols which have been already developed within the Spoken Dutch Corpus for PoS tagging, lemmatization and syntactic annotation but it will also explore the possibility of integrating an additional annotation layer based on semantic information. This annotation layer was not present in the Spoken Dutch Corpus.

The need for semantically annotated corpora has now become urgent. Several initiatives have been launched at the international level in the last years, showing that the time is ripe for activities in this direction. However, they have focussed almost entirely on English and not much attention has been dedicated to the creation of semantically annotated Dutch corpora. One of the goals of the D-Coi project is the development of a protocol for such an annotation layer. Only a small part of the corpus will be annotated with semantic information (i.e. 3000 words), in order to yield information with respect to its feasibility.[18]

[16] `http://taalunieversum.org/taal/technologie/stevin/`

[17] `http://lands.let.ru.nl/projects/d-coi/`

[18] The manual for semantic annotation (Monachesi and Schuurman 2006) plus the 3000 word corpus will be

A more substantial annotation effort could be carried out in the framework of the 500 million word corpus. In this follow-up project other types of semantic annotation might also be taken into consideration, as well as their interaction with other levels like PoS tagging and syntactic analysis. We are therefore taking this interaction into consideration when developing the protocols.

For the moment, we are only dealing with two types of semantic annotation and their interaction, that is semantic role assignment and temporal and spatial semantics. The reason for this choice lies in the fact that semantic role assignment (i.e. the semantic relationships identified between items in the text such as the agents or patients of particular actions), is one of the most attested and feasible types of semantic annotation within corpora. On the other hand, temporal and spatial annotation was chosen because there is a clear need for such a layer of annotation in applications like information retrieval or question answering.

## 5.2 Semantic role assignment in D-Coi

During the last few years, corpora enriched with semantic role information have received much attention, since they offer rich data both for empirical investigations in lexical semantics and large-scale lexical acquisition for NLP and Semantic Web applications. Several initiatives are emerging at the international level to develop annotation systems of argument structure, within the D-Coi project we intend to exploit existing results as much as possible and to set the basis for a common standard. We want to profit from earlier experiences and contribute to existing work by making it more complete with our own (language specific) contribution given that most resources have been developed for English.

The following projects have been evaluated in order to assess whether the approach and the methodology they have developed for the annotation of semantic roles could be adopted for our purposes:

- PropBank (Kingsbury, Palmer and Marcus 2002);
- FrameNet (Johnson, Fillmore, Petruck, Baker, Ellsworth, Ruppenhofer and Wood 2002);

Given the results they have achieved, we have taken their insights and experiences as our starting point. In the rest of this section, we will consider them more in detail in order to evaluate their strengths and weaknesses and to assess which features of the existing systems we want to include in the scheme for the semantic annotation of the D-Coi corpus.

available through the TST-centrale early 2007 (`http://www.tst.inl.nl/`).

### 5.2.1 PropBank

PropBank aims at adding a layer of semantic annotation to the Penn English TreeBank (Marcus, Santorini and Marcinkiewicz 1993). It provides a semantic representation of argument structures that are labeled consistently in such a way that the data are usable for automatic extraction. PropBank uses a very restricted set of argument labels.

The PropBank lexicon, which was added first to facilitate annotation and later evolved into a resource on its own, is constructed following a 'bottom-up' strategy: starting from the various senses of a word, a framefile is created for every verb. Such a framefile contains thus all possible senses of the verb plus a set of example sentences that illustrate the context in which the verb can occur. For each sense of the verb, a roleset and example sentences are available. Therefore, when a verb has two senses its framefile contains two different rolesets as is the case with *leave* from (Babko-Malaya 2005):

(3) a. Frameset leave.01 *move away from*
Arg0: entity leaving
Arg1: place left
Mary left the room

b. Frameset leave.02 *give*
Arg0: giver
Arg1: thing given
Arg2: beneficiary
Mary left her daughter-in-law her pearls in her will

To create a framefile, relevant sentences are extracted from the corpus. Based on those sentences, the most frequent and/or necessary roles are selected and one or more rolesets are formed. In this way, the most common senses of the verb are stored in the framefile. An interesting feature of the PropBank project is that the corpus has been annotated automatically with 83% accuracy and then corrected by hand on the basis of the developed lexicon. Furthermore, the goal of the project is to provide training data for supervised automatic role labelers. This is a desirable objective since it will be possible to annotate corpora of the size of D-Coi with semantic role information only if the the process is semi-automatic.

### 5.2.2 FrameNet

Contrary to PropBank, FrameNet does not annotate a complete corpus, but one that contains example sentences that illustrate all possible syntactic and semantic contexts of the lexical items taken into consideration. Besides the corpus, two other components can be distinguished in FrameNet, that is a set of lexical entries and a frame ontology.The development of the ontology is based on the frames. A frame represents a certain prototypical situation which is described by the frame definition. Every frame contains also a list of frame elements and a set of lexical units that can evoke the frame. The term *lexical unit* is used for a word in combination with one of its senses (Johnson et al. 2002). The frame elements fulfill a certain semantic role within the situation that is evoked by one of the lexical units. For every lexical unit a set of sentences is selected that illustrate all possible occurences of the lexical unit; all possible semantic roles are annotated in these sentences.

For example, the verb *leave* would evoke the frame *Departing* which is (partly) shown below:

- Departing
  An object (the Theme) moves away from a Source. The Source may be expressed or it may be understood from context, but its existence is always implied by the departing word itself.
- Frame Elements: Source, Theme, Area, Depictive, Distance, Manner, Goal etc.

A sentence annotated with semantic roles on the basis of the FrameNet information, would receive the following representation:

(4) $[_{Theme}$ We all $]$ left $[_{Source}$ the school$]$ $[_{Time}$ at four o'clock $]$.

Although FrameNet is still under development, its approach has been adopted for the annotation of semantic roles for languages other than English. An example is provided by the German project *Saarbrücken Lexical Semantics Annotation and analysis* (SALSA) (Erk, Kowalski, Pado and Pinkal 2003), but there are also projects based on FrameNet for languages such as Spanish, French and Japanese. However, SALSA distinguishes itself from the others by the fact that it is not restricted to building a lexicon but it annotates the complete German Tiger corpus using the FrameNet dictionary and adapting it to German. Unlike FrameNet, SALSA is not committed to always assigning a single sense (frame) to a target expression, or a single semantic role to a constituent but either more than one or an *Underspecified* sense tag can be assigned in case of vagueness or ambiguity.

### 5.2.3 Comparing approaches

The main differences we have noticed between FrameNet and Propbank are related to the methodology employed in the construction of the lexicon and the way the lexicon is structured. More generally, the classification attested in PropBank is based on *word senses* which are grouped in the 'shallow' framefiles while the FrameNet classification is driven by the *concepts* which are structured in the ontology of frames and thus based on hierarchically structured semantic classes.

Furthermore, the two projects differ with respect to the granularity of the role labels employed. FrameNet uses labels which immediately reflect the semantic role of the constituent and its annotation is rich in information. PropBank labels require more careful investigation about the meaning of the constituent in question.

The FrameNet labels are rather rich in information, however, they might not always be transparent for users and annotators. On the other hand, the advantage of the PropBank appoach is that by employing neutral labels, less effort is required from annotators to assign them. Furthermore, it creates the basis for the development of semi-automatic annotation of role labels, which is a necessary requirement if we want to annotate large corpora.

## 5.3 Merging approaches

In developing a scheme for the semantic annotation of the D-Coi corpus, we are faced with several options.

We could assume the FrameNet approach and develop a Dutch lexicon based on the English (and German) one and employ it for the annotation of the Dutch corpus. We would thus follow the strategy employed within the SALSA project and we could even exploit their results given the similarity between Dutch and German. A disadvantage of this choice is related to the fact that in order to annotate the corpus further we are bound to construct new frames (with their definitions and their frame entities) manually and this is a rather expensive process. Furthermore, we believe that the labels used to identify the frame entities are not very transparent and difficult for annotators to use.

The other possibility would be to employ the PropBank approach which has the advantage of providing clear role labels and thus a transparent annotation for both annotators and users. Furthermore, the annotation process could be at least semi-automatic. However, a disadvantage of this approach is that we would have to give up the classification of frames in an ontology which could be very useful for certain applications, especially those related to the Semantic Web.

Within the D-Coi project, we have chosen for a third option which wants to reconcile the rather pragmatic PropBank approach to role assignment which is essentially corpus based and syntax driven with the more semantic driven FrameNet approach which is based on a network of relations between frames. More generally, we would like to adopt the conceptual structure of FrameNet, but not necessarily the granularity of its role assignment approach. With respect to role assignment, we would like to adopt the annotation approach of PropBank. A risk we take is that we will end up with a semantic annotation layer which is too similar to the syntactic representation which is assumed in D-Coi. This will be an extension of that developed for the Spoken Dutch Corpus, that is a dependency structure which carries information about heads, complements and modifiers. However, a preliminary study carried out in (Stevens 2006), has shown that this is not the case and that the PropBank role labels provide additional useful information, especially with respect to modifiers. Stevens has suggested a heuristic strategy to map nodes in a D-coi dependency tree to PropBank argument labels. This strategy is implemented in a rule-based semantic role tagger (XARA), which has assigned 65% of the roles of the selected corpus correctly.

In order to assess the feasibility of our approach we have carried out a pilot study involving the integration of PropBank and FrameNet. The goals behind this study are:

- to assess whether it is possible to merge FrameNet frames with PropBank role labels and whether this merging has to be manual or whether it is possible to make it at least semi-automatic;

- to investigate to which extent we can use resources already developed for other languages;
- to assess whether we can extend existing resources on the basis of our language specific annotation and whether we should include the language specific features in the original resource;
- to investigate whether it is possible to extend the merged resources by exploiting the best features of both and in this way facilitate the process.

In our study we have considered a language independent phenomenon such as the classification of verbs of communication and a more language specific phenomenon, such as the classification of (adjunct) middle verbs. We refer to (Monachesi and Trapman 2006) for more details about the pilot study, in the rest of this section we will exemplify the merging approach on the basis of the *Communication* frame.

### 5.3.1 Merging approaches: The Communication frame

As previously discussed, FrameNet provides a rich semantic representation of language because its lexicon not only encodes word senses but also relations among words. Words can be related to each other on the basis of the frame they share, but also because a relationship among frames is established. The ontological relations add extra information to word senses. In FrameNet, every frame has its own definition which distinguishes it from other frames while the frame elements can be the same across frames due to (partial) inheritance. However, there is a great variety of elements that are frame specific creating thus a quite complex structure which is not always very transparent for annotators and users. Furthermore, by taking into account the *Communication Frame*, which comprises a mother frame with six daughters, we have noticed that the inheritance relation is not as strict as we had assumed. Our aim is thus to reduce the FrameNet frames to a simpler form in which the set of frame elements is restricted to a number of elements that is comparable with the PropBank arguments. Since the interpretation of a PropBank argument label depends on the word senses of the individual word, we wanted to make their interpretation more uniform as well. This can be achieved by assuming Levin's classes and diathesis alternation and the revisions implemented within VerbNet . Verbs within the same Levin class, sharing the same diathesis alternations, should have the same roleset. Thus, the second step is to group together those verbs that share the same FrameNet frame, the same Levin class and diathesis alternation and assign this group one roleset; this roleset is derived from PropBank by selecting the most common arguments from one group of verbs. Regrouping the verbs this way decreases the number of rolesets in comparison with the number of PropBank rolesets. The advantage is that we can determine rolesets using a simple algorithm that results in an intersection of the FrameNet and the Levin classification. Assigning rolesets to these newly created classes takes less effort than manual role assigment for every individual verb. We then compared this roleset with the frame elements that are normally used.

As a test case we took the frame *Communication* which has six daughters:

*Communication-manner, Communication-noise, Communication-response, Gesture, Reassuring and Statement*. For example, the verbs comprised in *Communication-noise* belong to four different Levin classes with the majority of the verbs belonging to the class *Verbs of Manner of Speaking*. These are verbs like *babble, bellow, croon, hiss, wail, whine* etc. A general roleset for this group could be:

| PropBank | FrameNet |
|---|---|
| Arg0: speaker, communicator | Speaker |
| Arg1: utterance | Message |
| Arg2: hearer | Addressee |

TABLE 3 Roleset for *Communication-Noise*

The alignment in the case of the communication verbs is rather straightforward and it seems to suggest that indeed this methodology might be appropriate.

## 5.4 Temporal semantics

### 5.4.1 Background

The layer of temporal and spatial annotation is meant to be useful for both scientific research as well as applications (information retrieval, question answering, multidocument summarization, etc.). Within the STEVIN-programme this layer of annotation is part of a whole series of annotations, from part-of-speech (or morphosyntax) over syntax to several semantic ones. It goes without saying that it is to reflect the state of the art. In that respect TimeML (Saurí, Littman, Knippen, Gaizauskas, Setzer and Pustejovsky 2006) comes to mind as far as temporal annotation is concerned.[19]
TimeML is a temporal markup language, a joint effort reflecting many ideas from other, earlier approaches (it is strongly based on an earlier version of TIDES ((Ferro, Gerber, Mani, Sundheim and Wilson 2002)) as well as on (Setzer 2001) whose authors were among the developers of TimeML, the main difference being that more types of phenomena are annotated, especially those related to events and to tense and aspect). But note that also for TIDES there is an adapted version (TIDES 2005) of their manual, still concentrating on the core time expressions, so-called timexes (such as calendar dates). Within D-Coi we wanted to annotate states and events as well, cf. TimeML.

TimeML is designed as a common meta-standard for temporal annotation covering the recognition of all temporal elements (i.e. expressions and events (for the latter notion, see pt 2 below)), anchoring of these elements and relating them to each other.
There are four meta data structures to be annotated, cf. (Day, Ferro, Gaizauskas, Hanks, Lazo, Pustejovsky, Saurí, See, Setzer and Sundheim 2003):

[19] In this paper we will concentrate on temporal annotation. Spatial annotation is done in a similar way, concentrating on spacexes instead of timexes.

- Events: these describe all situations that occur or happen. States are considered to be events, only a subset of these will be annotated.
- Times (Timex3): points, intervals or durations. These may be referred to by fully specified temporal expressions (like *May 4th, 2005*), by underspecified expressions (*Monday*), contextually dependent expressions (*last week*), ...
- Signals: elements (like prepositions, conjunctions) indicating how temporal objects are to be related.
- Links: these describe the relations between events, and between events and times or signals. There are three kinds of links: temporal links (like `before, immediately after, included in`), subordination links (like `modal, negative, factive`), and aspectual links (like `initiation, continuation`).

TimeML is by far the most elaborated annotation scheme around these days, there is also a still rather small corpus available (TimeBank) that is annotated according to the TimeML guidelines.

There are, however, a few problems when adopting (and adapting) a scheme like TimeML for the Flemish/Dutch STEVIN programme:

1. we want to make use of information available through other layers like Syntactic Analysis (SA) and Part of Speech tagging (PoS) when analysing the sentences
2. the semantic foundation should be a sound one
3. the annotation should be useful for the scientific community
4. annotation should be feasible in a semi-automatic way although we want to annotate all sentences in a text, i.e. also those without so-called timexes.

With respect to point 1, like most annotation schemes around TimeML does start from scratch, not really taking into account other annotation layers (at least not in a way a script can be aware of it). Within D-Coi we wanted to make use of all information available (such as Part of Speech, Syntactic Analysis)
The issue under 2 is of a more serious nature. In TimeML, states are considered particular types of events, which is not correct: they are at the same level, and they both belong to the 'eventualities' (or 'situations').[20] The other types of 'events' used in TimeML are also not standard ones, cf. above. We therefore will not make use of this part of TimeML, although we do see the merits of a characterization of verbs in order to rate the relevance of a temporal expression. We are using a separate feature to accommodate this.
Point 3 is related to point 2: an elaborated tense and aspect component is often not considered necessary for applications, especially when the corpus to be annotated consists of news items, cf. (Setzer 2001). We wanted to make use of a more elaborate theory of tense and aspect than the one used in TimeMl as this is of importance for temporal semantics (as opposed to temporal annotation), the more as we have to annotate all kinds of texts,

[20]Cf. also (Mani, Pustejovsky and Gaizauskas 2005), p. 491.

that is not only news items, but also fiction and the like. We therefore want to merge the ideas behind TimeML (and TIDES) with those of theories like Discourse Representation Theory (Kamp and Reyle 1993). This is in fact our most serious content-related objection to TimeML.[21]
The last point seems to be contradictory: we want to annotate more phenomena, and at the same time we want to do it in a semi-automatic way, whereas other annotation schemes seem to rely heavily on a firm amount of manual annotation. We do so by making use of compositionality, exploiting all the regularities in the language. We also need to annotate full texts, not isolated sentences.[22]

## Our approach

As remarked in the previous section, in our approach we cover more or less the same phenomena as in TimeML: we annotate all temporal expressions (nouns, prepositions, adverbs, adjectives, conjunctions, as well as the relations between them); eventualities (events, states and processes); characterization of eventualities as reporting, perception etc in order to be able the reliability of an statement (compare: *Bill Gates is CEO of Microsoft* vs *Bill Gates claims/is said to be CEO of Microsoft*); tense and aspect.

We exploit lexica with temporal items (lemmata, sometimes with a very specific PoS label in order to be able to refer to a 'token'; expressions). In order to give an idea how entries would be combined, have a look at the following table with the temporal semantic information attached to the leaf nodes of the temporal expression *23 maart 1967 om twintig na drie* (the 23rd of March 1967 at 20 minutes past three).

| | |
|---|---|
| 23 | t-ent="yes" t-value="D23" |
| maart | t-ent="yes" t-value="M03" |
| 1967 | t-ent="yes" t-value="Y1967" |
| om | t-ent="yes" mod="at" |
| twintig | t-ent="yes" t-value="T20M" |
| na | t-ent="yes" mod="after" |
| drie | t-ent="yes" t-value="T03\|15H" |

In combination this will become **1967-03-23T03|15:20**.[23] In case we would have known that it would be *drie uur 's middags* (three o'clock in the afternoon) the full expression would get the value **1967-03-23T15:20**. In the first expression the value of the hour is left underspecified.

There will also be a lexicon containing those expressions that at first sight seem to be temporal as they contain an item that usually is used in a temporal way: *Zwarte September*

[21]Within the framework of this paper it is not possible to give a detailed overview of our approach as far as tense and aspect is concerned. We refer the interested reader to the manual (Monachesi and Schuurman 2006).

[22]Note that for phenomena like coreference we rely on a project like COREA, another STEVIN-project, to solve these computationally.

[23]We will use | as a symbol meaning 'or'.

(Black September), *De Morgen* (a Flemish newspaper), *een dagje ouder worden* (be getting on a bit), *ouden van dagen* (elderly people). Such expressions will be excluded from temporal annotation. Note that some expressions are ambiguous in this respect: *Het is vijf voor twaalf* (a) It is five to twelve; b) We are on the verge of disaster). In such a case the broader context will be dicisive.
On the other hand, one also needs a lexicon containing expressions that do get a temporal interpretation although they don't contain temporal items, like *Tweede Wereldoorlog* (Second World War).

As said above, up till now the only annotation scheme in which events (or rather eventualities) get an inclusive treatment , cf. (Mani et al. 2005) (but see above) is TimeML.

But also when dealing with timexes (thus neglecting eventualities) some problems arise. In (Mani et al. 2005) three types are mentioned as problematic:

1. indexicals: contextual dependent expressions, like *Wednesday* (which Wednesday?) or *next week*
2. relational expressions: times are specified in relation to other times *two weeks after Christmas*
3. vagueness: times with inherently vague boundaries *spring, evening*

In the next section we will say more about the third category (vague expressions) as they occur in general language, and the way we deal with such expressions.

**Vague expressions**

When reasoning with time, for example to answer a *when*-question, one is confronted with an annoying human characteristic: people tend to use their language in a very sloppy way.
It is therefore sometimes rather 'dangerous' to deduce temporal information and to reason with it (but see (Pan, Mulkar and Hobbs 2006).) There are several ways of being sloppy when temporal information is concerned. We will try to accommodate these in various ways.

Case A:

Suppose today is Friday, March 10, 2006. When refering to Saturday the 18th in the Netherlands one will use the expression in (5), in Flanders the one in (6):

| | | | | |
|---|---|---|---|---|
| (5) | morgen | over | een | week |
| | tomorrow | over | a | week |

a week from tomorrow

| | | | | |
|---|---|---|---|---|
| (6) | morgen | over | acht | dagen |
| | tomorrow | over | eight | days |

a week from tomorrow

The Flemish expression *over acht dagen* will be in the lexicon with the meaning *over* **zeven** *dagen*.[24]

The following case is more serious:

Case B:

Suppose it is April 18th 2006, the day after Easter Monday. That Tuesday you can mention that in 14 days time there is already another public holiday, Labour day (Monday 1st). Nobody will correct you, saying that it should be '13 days'. One should have protested in case you did say *in exactly 14 days*.
*14 days* or *2 weeks* are a kind of rather global containers, meaning *more or less 14 days/2 weeks*, whereas for example *12 days* or *16 days* only have a strict meaning.
In order not to jump to false conclusions (like: Labour Day is May 2nd) we use a boolean feature *noise*. That way we would still conclude that Labour Day is May 2nd, but with the warning that this can be wrong. Note that in a case like this a human corrector will correct the mistake as everybody knows that the reference is to May 1st. The point however is that such sloppy (temporal) containers are used time and again. In case of expressions like *14 dagen*, we add the feature noise="yes" which expressions like *16 dagen* will not have.

Case C:

There is yet another type of global temporal expressions, those in which the uncertainty is made explicit in the wording:

(7) Het was ongeveer middernacht toen ...
It was more or less midnight when ...
It was about midnight when ...

(8) Het was rond middernacht toen ...
It was around midnight when
It was around midnight when ...

In these cases we will use mod="approx", in order to modify the value T24:00, cf. (Ferro, Gerber, Mani, Sundheim and Wilson 2005) and (Saurí et al. 2006). Note that in case of

(9) Het liep tegen middernacht
It got on towards midnight
It was getting on towards midnight

(10) Het was net na middernacht
Het was net na middernacht

[24]Note that in French the expression is *quinze jours* (fifteen days) instead of *two weeks*.

It was just after midnight

the value of the modifier will not be just “approx”, but the more specific “just-before” resp. “just-after”.

Case D:

The seasons of the year are also often used in a sloppy way, refering globally to those particular months. In most annotation schemes they are annotated as SP, SU, FA or WI respectively (for ’spring’, ’summer’, ’fall’ and ’winter’). Within D-Coi we refer to the seasons with months as we want to be able to order eventualities as in 11:

(11)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| De | ring | rond | Antwerpen | wordt | deze | zomer | vernieuwd. | Eind | Mei |
| The | ring road | around | Antwerp | will be | this | summer | renewed. | End | May |

| | | | |
|---|---|---|---|
| wordt | de | Singel | aangepakt. |
| will be | the | Singel | dealt with. |

The ring road around Antwerp will be renewed this summer. The end of May the Singel will be dealt with.

A human annotator will know that the Singel is likely to be renewed before the ring way, as May is in the spring, and therefore before the summer. A machine will not know that unless it is specified. As May has the value M05, and summer M07/09[25] the machine does know that May is ordered before summer. Of course end of May next year could have been meant, but in that case this would have been said so explicitly.

Notions like *meteorologische zomer* of *weerkundige zomer* (meteorological summer) and *astronomische zomer* (astronomical summer) are as such part of the lexicon with complex entries.

The noise feature is added because people tend to use the names of the seasons in a sloppy way, for example influenced by the weather, as they are not aware of the exact dates at which the seasons change.

Case E:

This is in fact the case we mentioned in section 5.4.1 in expressions like *23 maart 1967 om twintig na drie* (the 23rd of March 1967 at 20 minutes past three). With respect to *drie uur* there are two options: T03 or T15. In case the context doesn’t make it clear which one is meant, we will take “T03|15” as its value.

[25]The / is used as a symbol meaning ’up to and including’.

| | type | solution |
|---|---|---|
| A | *morgen over acht dagen* | lexicon |
| B | *over twee weken* | noise="yes" |
| C | *ongeveer middernacht* | mod="about" |
| D | *de herfst* | t-value="M09/12" noise="yes" |
| | *in de ochtend* | t-value="T05/13" noise="yes" |
| E | *om drie uur* | t-value="T03\|15" noise="yes" |

## 5.5 Integration of annotation schemas in D-Coi

As already mentioned, within the D-Coi project a choice has been made, with respect to which types of semantic annotation should be developed: annotation of semantic roles as well as of temporal and spatial semantics. At the moment, we keep the two annotation levels separate, to make it easy to produce alternative annotations of a specific type of semantic information without need to modify the annotation at the other level. By keeping the different types of annotation separately, it will be possible to enable progress on techniques for one type of semantic processing without need to wait for the development of high-performing systems for other aspects of semantic interpretation. However, we are aiming at a comprehensive annotation scheme which should ensure compatibility among the various types of semantic information.

Since all linguistic levels interact closely in order to determine the meaning of a whole sentence, the meaning of an expression will be characterized not only by its word meanings, but also by the manner in which they are put together: syntactic structure plays thus a relevant role. In the D-Coi project, the two different types of semantic annotations will be carefully integrated with the other layers of annotation, that is syntactic and morphosyntactic. Allowing semantic annotation to proceed in parallel with the other levels of annotation is a great advantage. There are several examples of treebanks which were extended with semantic information at a later stage such as PropBank or the Prague Dependency Treebank. While these additions are possible, they are not trivial since they often require modifications in the previous annotations, such as changing the labels or some design principles. With D-Coi, we are in the privileged position of developing these annotations in parallel, taking thus into account possible interactions and being able to exploit the available information. The guidelines developed in this respect can constitute the basis for further research as well as a reference for similar initiatives.

In particular, our input sentences are syntactically analysed in another layer using the Alpino parser,[26] in this way the meaning of an entity (expression, sentence) will not only be characterized by the meaning of the constituting words, but also by the manner in which these are put together. Note that for example in temporal semantics the interaction of a verb and other constituents (especially their prepositional and/or nominal heads) is crucial in order to decide whether the verb is refering to a bounded or an unbounded event. PoS information is also still available at the syntactic level, for example with respect to

[26] `http://odur.let.rug.nl/˜vannoord/alp/`

temporal information associated with the verbal forms. Similarly, in the case of semantic role labelling, the syntactic structure encoded will guide the assignment of the role labels, this is the case in the distinction between arguments and modifiers.

### 5.5.1 Conclusion

Our work on both types of semantic annotation discussed in this paper shows that it is feasible to annotate a rather substantial corpus with this kind of information as well, for example a subset of the 500-million-word corpus mentioned in section 1. Our annotation is designed in a way that other semantic layers, like coreference, negation, lexical sematics, can be added as well.
Such a project fits very well in the international state of affairs, cf. recent efforts in the States by Hovy (cf. his keynote lecture at CLIN 2005, Amsterdam) and Pustejovsky (Pustejovsky, Saurí and Littman 2006) and their groups.

## References

Babko-Malaya, O.(2005), *Guidelines for Propbank framers*.

Day, D., Ferro, L., Gaizauskas, R., Hanks, P., Lazo, M., Pustejovsky, J., Saurí, R., See, A., Setzer, A. and Sundheim, B.(2003), The TimeBank Corpus, *Corpus Linguistics 2003*, Lancaster.

Erk, K., Kowalski, A., Pado, S. and Pinkal, M.(2003), Towards a Resource for Lexical Semantics: A Large German Corpus with Extensive Semantic Annotation, *Proceedings of ACL 2003*.

Ferro, L., Gerber, L., Mani, I., Sundheim, B. and Wilson, G.(2002), *Instruction Manual for the Annotation of Temporal Expressions*, MITRE Washington C3 Center, McLean, Virginia.

Ferro, L., Gerber, L., Mani, I., Sundheim, B. and Wilson, G.(2005), *TIDES 2005 Standard for the Annotation of Temporal Expressions*.

Johnson, C., Fillmore, C., Petruck, M., Baker, C., Ellsworth, M., Ruppenhofer, J. and Wood, E.(2002), FrameNet: Theory and Practice.

Kamp, H. and Reyle, U.(1993), *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, Vol. 42 of *Studies in Linguistics and Philosophy*, Kluwer Academic Publishers, Dordrecht, Boston, London.

Kingsbury, P., Palmer, M. and Marcus, M.(2002), Adding Semantic Annotation to the Penn Treebank, *Proceedings of the Human Language Technology Conference. HLT-2002*.

Mani, I., Pustejovsky, J. and Gaizauskas, R. (eds)(2005), *The Language of Time. A Reader*, Oxford University Press.

Marcus, M., Santorini, B. and Marcinkiewicz, M.(1993), Building a large annotated corpus of English: The Penn treebank, *Journal of Linguistics*.

Monachesi, P. and Schuurman, I.(2006), *Semantic Annotation for D-Coi. A manual*, Universiteit Utrecht and Katholieke Universiteit Leuven. version 0.1.

Monachesi, P. and Trapman, J.(2006), Merging FrameNet and PropBank in a corpus of written Dutch, *Proceedings of LREC 2006*.

Oostdijk, N., Goedertier, W., Van Eynde, F., Boves, L., Martens, J., Moortgat, M. and Baayen, H.(2002), Experiences from the Spoken Dutch Corpus Project, *Proceedings of LREC 2002*.

Pan, F., Mulkar, R. and Hobbs, J.(2006), An Annotated Corpus of Typical Durations of Events, *Proceedings of LREC 2006*, Genoa.

Pustejovsky, J., Saurí, R. and Littman, J.(2006), Argument structure in TimeML, *Workshop on Merging and Layering Syntactic Information*, LREC, Genoa, Italy.

Saurí, R., Littman, J., Knippen, B., Gaizauskas, R., Setzer, A. and Pustejovsky, J.(2006), *TimeML Annotation Guidelines, version 1.2.1*.

Setzer, A.(2001), *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study*, PhD thesis, University of Sheffield.

Stevens, G.(2006), *Automatic semantic role labeling in a Dutch corpus*, Master's thesis, University of Utrecht.

6

# A Pilot Study for a Corpus of Dutch Aphasic Speech (CoDAS)

## Focusing on the orthographic transcription

ELINE WESTERHOUT AND PAOLA MONACHESI

*Utrecht University, Uil-OTS*
*Trans 10, 3512 JK Utrecht, The Netherlands*
{*Eline.Westerhout, Paola.Monachesi*}*@let.uu.nl*

## Abstract

In this paper, a pilot study for the development of a corpus of Dutch Aphasic Speech (CoDAS) is presented. Given the lack of resources of this kind not only for Dutch but also for other languages, CoDAS will be able to set standards and will contribute to the future research in this area. We have established the basic requirements with respect to text types, metadata, and annotation levels that CoDAS should fulfill. Given the special character of the speech contained in CoDAS, we cannot simply carry over the design and annotation protocols of existing corpora, such as the Spoken Dutch Corpus (CGN) or CHILDES. However, they have been taken as starting point. We have investigated whether and how the procedures and protocols for the orthographic transcription and the part-of-speech tagging used for the CGN should be adapted in order to annotate and transcribe aphasic speech properly.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands.*
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

## 6.1 Introduction

The Corpus Gesproken Nederlands ('Spoken Dutch Corpus', CGN) (Oostdijk, Goedertier, van Eynde, Bovens, Martens, Moortgat and Baayen 2002) represents an important resource for the study of contemporary standard Dutch, as spoken by adults in the Netherlands and Flanders. However, it only contains speech from adults with intact speaking abilities. There is the need to develop specialized corpora that represent other types of speech. The JASMIN project has already been dedicated to extending the CGN with speech of elderly people, children and non-natives (Cucchiarini, van Hamme, van Herwijnen and Smits 2006). In our project, we performed a pilot study for the development of a corpus containing aphasic speech: CoDAS, a Corpus of Dutch Aphasic Speech (Westerhout 2006). In this study, we have established the basic requirements with respect to text types, metadata and annotation levels that this corpus should fulfill. Furthermore, we have investigated the challenges that aphasic speech poses for orthographic transcription and part-of-speech tagging.

Given the special character of aphasic speech, we cannot simply carry over the design and the annotation protocols of existing corpora, such as CGN or CHILDES (MacWhinney 2000). However, they have been taken as starting point. For the orthographic transcription, the phonetic transcription and the part-of-speech tagging, we have investigated whether and how the existing procedures and protocols written for the annotation and transcription of the CGN could be adapted in order to make them suitable for the annotation and transcription of aphasic speech. In this paper, we focus on the adaptation of the orthographic transcription protocol of the CGN.

## 6.2 Aphasia

The abilities to understand and produce spoken and written language are located in multiple areas of the brain (i.e. in the left hemisphere). When one of these areas or the connection between them is damaged, the language production and comprehension becomes impaired. This language impairment is called "aphasia". In the Netherlands, about 30,000 people suffer from aphasia. In 85% of the cases, the cause of aphasia is a CVA (stroke). Other causes are traumatic brain injuries (12%) and brain tumors (3%) (Davidse and Mackenbach 1984).

Language impairments differ depending on the location and size of the damage. As a consequence, different aphasia varieties can be distinguished. The main varieties are Broca's aphasia, Wernicke's aphasia, and global aphasia. Individuals with Broca's aphasia frequently speak in short, meaningful phrases that are produced with great effort. Broca's aphasia is thus characterized as a nonfluent aphasia. Function words such as *is*, *and*, and *the* are often omitted. Individuals with Wernicke's aphasia may speak in long sentences that have no meaning, add unnecessary words, and even create new "words". Persons suffering from global aphasia have severe communication difficulties and will be extremely limited in their ability to speak or comprehend language.

However, most aphasia patients do not neatly fit into one of the existing categories. Their speech bears characteristics of different types of aphasia. For the purpose of our investigation, it was sufficient to distinguish between fluent and nonfluent aphasia. Nonfluent aphasia is characterized by heavy syntactic disorders in which inflectional affixes and function words are often missing whereas in fluent aphasia the syntax is not the main problem, but language comprehension and language repetition are impaired. The patients participating in the pilot study were all suffering from nonfluent aphasia.

## 6.3 Corpus Design

CoDAS can become an indispensable tool for research on aphasia since it will offer a considerable amount of speech data. Collecting data is a very time consuming enterprise due to the language impairment of the patients and privacy issues and IPR (section 6.3.1). It is for this reason that each researcher gathers his own data and is not allowed to share it. CoDAS can change this state of affairs since the data included in the corpus could be made accessible to all researchers. The corpus will be relevant not only for research on language and speech processing, but also for the development of real life speech applications and for the creation of programs for diagnosing patients. Speech and language therapists could also benefit from it.

Given the lack of resources of this kind not only for Dutch but also for other languages, CoDAS will be able to set standards and it will contribute to the future research in this area. Therefore, the corpus should fulfill at least the following requirements which will be discussed in more detail in the rest of this section. First, it should constitute a plausible sample of contemporary Dutch spoken by aphasic patients. Important issues are the inclusion of the different aphasia varieties and various communicational settings (section 6.3.2). Second, the speech fragments have to be well-documented with metadata about the aphasic speakers (section 6.3.3). Finally, the corpus should be enriched with linguistic information, such as part-of-speech tags, syntactic and prosodic annotation, as well as phonetic transcription (section 6.3.4).

### 6.3.1 IPR

As already mentioned, one of the problems related to the collection of aphasic speech data is the fact that obtaining permission for recording and distributing data from aphasic patients is not straightforward. Even if aphasic speakers give researchers permission to record their speech and to make it available to others, this does not automatically permit public access to their speech data. In the Netherlands, the Medical Ethics committee has to grant permission for public access to their speech.

Ideally, we would like CoDAS to include authorized access to the original recordings. In case the permission for including the recordings cannot be obtained, it is important that the transcriptions are as detailed as possible. Except for privacy information, everything should be represented in the transcriptions.

### 6.3.2 Text types

CoDAS should encode a plausible sample of contemporary Dutch as spoken by aphasic patients, that is it should include speech representing different types of aphasia (Broca, Wernicke, global, transcortical, anomic, etc.) as well as various communication settings. Interviews between a nonaphasic person and an aphasic person such as the ones carried out in the context of the Aachen Aphasia Test (AAT) could be included. Other subtests of the AAT can also be used. Conversations of the aphasic patients at home, and in aphasia centers will also be useful text types. (Westerhout and Monachesi 2006) give more information on possible text types that could be included.

### 6.3.3 Metadata

Metadata play an important role in enhancing the usability of the collected data, for example they can be used to define and access precisely those subsets of data that are relevant for the user. However, because of the special character of the corpus of aphasic speech, not only general information about the patients needs to be collected (e.g. age, gender, place of residence) but also some more specific features. For example: time post-onset (how long has the patient been aphasic at the time of speaking), cause of aphasia, paralysis (aphasia can be accompanied by paralysis of one or more parts of the body, most times the right part of the body is paralyzed), handedness, verbal apraxia (articulation disorder as a result of problems in planning the articulation movements), dysarthria (a speech impairment as a result of a neurological disorder), type of aphasia, and severity of aphasia (according to the AAT).

### 6.3.4 Annotation and transcription

As in other corpora, orthographic transcription is required in a Corpus of Aphasic Speech because it serves as basis for all other annotation and transcription levels.

Depending on the research questions to be answered, phonetic transcription can also be relevant. Aphasic patients often make phonetic or phonological errors and frequently encounter articulation problems. The phonetic annotation can provide users with information about these errors which would not be accessible via the orthographic transcription, that makes use of standard spelling conventions. Ideally, speech and video recordings should be attached to the transcription in order to be able to listen and watch the fragments on request. Video recordings can be helpful, because aphasic patients sometimes use gestures to explain what they mean and as a strategy to find words. As a first step, a grapheme-to-phoneme converter can be used to perform the phonetic transcription automatically. This automatically created transcription has to be corrected manually (Binnenpoorte 2006).

Information about part-of-speech should be provided since it can shed light on questions about the word classes which are typically left out by patients. Researchers might be interested in, for example, the number of used verbs, finiteness of the verbs, used determiners, the relation between determiners and finiteness, the number of pronomina, etc..

The part-of-speech tagging can be performed automatically. For the tagging of Dutch text several taggers are available (Zavrel and Daelemans 1999). However, existing taggers need to be adapted in order to produce a reasonable level of accuracy of aphasic speech annotation (Section 6.4.4).

Syntactic annotation should also be included in a Corpus of Aphasic Speech since aphasia often influences the syntax of speech. Several parsers are available for the syntactic annotation of Dutch texts, however, also in this case they have to be adapted to be able to deal with ungrammatical sentences, uncomplete sentences and sentences with mirror constructions.

The prosody of nonfluent aphasic patients is often damaged because of the efforts the patients make in the production of speech. Just as for the phonetic transcription, it will be better to have the speech and video recordings attached to the transcriptions.

## 6.4 The pilot study

A pilot study has been carried out to investigate to which extent existing annotation and transcription protocols already developed for corpora such as CGN or CHILDES could be adopted for the setup of CoDAS. To this end, speech material of aphasic patients has been collected and annotated on the basis of the existing protocols which have been revised accordingly.

### 6.4.1 Patients

Speech material of six aphasic patients has been collected. The average age of the patients was 54 and the time post onset was between three and four years. The six patients could not be assigned to one variety according to the AAT, which was conducted by a qualified Speech and Language Pathologist. However, they were all diagnosed as having a nonfluent aphasia according to this test. To determine the fluency, the sixth score on the Spontaneous Language Sample subtest indicating the syntactic structure has played a major role. The results on the subtest ŠSpontaneous Language SampleŠ of the AAT were used as speech samples for the pilot study. Table 4 shows the results on the complete AAT and on the subtest 'Spontaneous Language Sample' for the six patients.

### 6.4.2 Relevant corpora

Two corpora have been of particular relevance for our pilot study and have been used as starting point for the definition of the transcription and annotation protocols, that is the CHILDES corpus and the CGN.

The CHILDES corpus is important because the kind of speech which has been collected within this project also deviates from ŠnormalŠ speech. It contains mainly speech data of young monolingual (normally developing) children interacting with their parents or siblings, but there is a small part with transcripts of children with language disorders (e.g.

| | Patient | | | | | | |
|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | |
| Spontaneous speech sample | 3 | 3 | 3 | 2 | 4 | 3 | (COM) |
| | 4 | 4 | 4 | 4 | 5 | 4 | (ART) |
| | 4 | 4 | 5 | 4 | 5 | 5 | (AUT) |
| | 3 | 3 | 4 | 3 | 4 | 5 | (SEM) |
| | 3 | 3 | 3 | 3 | 4 | 4 | (PHO) |
| | 2 | 2 | 2 | 1 | 2 | 2 | (SYN) |
| Percentage Aphasia | 100 | 100 | 100 | 100 | 98.4 | 86.5 | |
| Percentage Broca | 14.3 | 47.4 | 69.1 | 99.9 | 8.1 | 47.0 | |
| Percentage Wernicke | 26.2 | 52.6 | 30.8 | 0.1 | 21.4 | 1.2 | |
| Percentage Anomic | 59.5 | 0 | 0 | 0 | 70.5 | 51.8 | |
| Aphasia type | ? | ? | ? | Broca | Amnestic | ? | |

TABLE 4 The scores on the AAT of the patients involved in the pilot study

Down syndrome, autism), bilingual children, second-language learning adults, and aphasics. The CHILDES manual (MacWhinney 2000) presents coding systems for phonology, speech acts, speech errors, morphology, and syntax. The user can create additional coding systems to serve special needs. The CHILDES guidelines have been a reference for the development of the protocols which will be used in the annotation of CoDAS.

The second corpus of interest in our pilot study is the CGN given that it is also a corpus of spoken Dutch. The CGN is a database of contemporary standard Dutch as spoken by adults in the Netherlands and Flanders. The corpus comprises approximately ten million words (about 1,000 hours of speech), two thirds of which originates from the Netherlands and one third from Flanders. It contains a large number of speech samples recorded in different communicational settings. The extensive protocols written for the different transcription and annotation levels of the CGN were used as starting point for the pilot study.

### 6.4.3 Orthographic transcription

Transcribing spontaneous speech is quite complicated, because it is not fluent and contains filled pauses, mispronunciations, false starts, and repetitions. Besides, it is often difficult to distinguish utterance boundaries. For the transcription of the aphasic speech data the protocols used for the transcription of the CGN and CHILDES have been used and were adapted to make them suitable for the transcription of aphasic speech.

#### CGN and CHILDES

The orthographic transcription protocol of the CGN is based on the EAGLES guidelines developed for the transcription of spontaneous speech. The protocol is based on three criteria, which were kept in mind while adapting the protocol to make it suitable for the orthographic transcription of aphasic speech. The three criteria underlying the orthographic transcription protocol of the CGN are (Goedertier, Goddijn and Martens 2000):

- Consistency: in order to increase consistency, standard spelling conventions are maintained. However, in a number of cases it is necessary to deviate from standard conventions to transcribe accurately what has been said. For example, when a word is not finished, only the part of the word that has been uttered should be transcribed. For indicating such problematic issues special symbols were defined.
- Accuracy: to improve the quality of the transcriptions, all orthographic transcription files were checked by a second transcriber
- Transparency: the number of transcription rules are kept down to a minimum. This makes it easier to memorize and apply them.

The guidelines for the orthographic transcription of the CGN and CHILDES are both almost entirely based on the EAGLES guidelines. However, at some points complementary guidelines are required to deal with typical Dutch phenomena. Besides, for the transcription of non-speech acoustic events (such as coughing and relevant background noise) guidelines are needed. So the guidelines can be divided into three groups: 1) spelling guidelines corresponding to the EAGLES guidelines, 2) complementary spelling guidelines, 3) guidelines for dealing with non-speech material (e.g. coughing, not finished words).

## Spelling guidelines corresponding to the EAGLES guidelines

**Reduced word forms:** For the CGN, the lexicon contains the most common reduced forms (e.g. *'k* for *ik* ('I'), *da's* for *dat is* ('that is')). The orthographic transcribers have to use the forms that are on this list when they are heard instead of the full forms. In the CHILDES project, parentheses are used to deal with this phenomenon. The sounds that are dropped are shown between brackets, so when a transcriber hears *bout* instead of *about* he transcribes *(a)bout*. For the transcription of Dutch abbreviations, the same procedure has been followed as for the English shortened forms. When a person says *es* instead of *eens* (*just*) this is transcribed as *e(en)s*.

**Dialect forms:** Dialect words and constructions that do not consist in standard Dutch but are of a rather dialectal nature are followed by *d according to the CGN protocol. Besides typical dialect words, such as *keuje*d* for *varken* ("pig"), there are several constructions that are typical for a specific region. An example of such a dialectal construction is the inflection of articles, pronouns, adjectives, and substantives in the South of the Netherlands (e.g. *nen*d blauwen*d auto* for *een blauwe auto* ("a blue car")). Words that belong to standard Dutch, but are pronounced dialectically, are followed by *z (e.g. *jou* ("you") pronounced as /ju/ is transcribed as *jou**z). The CHILDES system lets annotators choose one out of four options for annotating dialect forms. The four possibilities are (1) Adding each variant to the lexicon file; (2) Adding the standard form after each variant form; (3) Creating a full phonological transcription of the whole interaction and linking this to an audio file; or (4) Ignoring dialectal variation and transcribing the standard form.

**Numbers:** For both the CGN and the CHILDES system, numbers have to be written out in words. When a number can be pronounced in more ways, the number should be transcribed in the way it is pronounced (e.g. 1837 can be either *achttienhonderd zevenendertig* ("eighteen hundred thirty-seven") or *achttien zevenendertig* ("eighteen thirty-seven")). Within the CGN, there also is a second option for transcribing numbers: the numbers 0 up to and including 99, the hundreds, thousands and hundred thousands can also be written as figures (e.g. 1837 can be either 1800 37 or 18 37). These numbers will then be converted automatically into the written form.

**Abbreviations and spelled words:** In the CGN, spelled words or separate letters are written in capital letters (e.g. *laf* ("cowardly") becomes L A F). When letters are spelled in an alternative way, they are not written in capital letters but in the way they are pronounced, *u is assigned to each separate letter (e.g. *laf* can also be "spelled" as le*u a*u fe*u). Abbreviations do not get a special symbol and are written in the way they are used. When the component letters of a abbreviation are pronounced separately (e.g. as in *t.z.t.* ("in due time")), they are written in capital letters as one word, without white spaces between the component letters (e.g. *BTW* for *BTW* ("VAT"), *TZT* for *t.z.t.*). Acronyms are written in the way the standard spelling prescribes, but always completely in capital letters (e.g. *NASA*, *TROS*).

The CHILDES guidelines differ slightly from the CGN guidelines. For words that are spelled out each separate letter gets the symbol @l (e.g. *word* becomes *w@l o@l r@l d@l*). Acronyms are transcribed by using the component letters as a part of a linked form, the @l marking is not used for acronyms (e.g. *USA* becomes *U_S_A*). Acronyms that are not spelled out when produced are written as words (e.g. *Benelux*). Abbreviations for titles are also written out in their full form (e.g. *Mister* instead of *Mr.*).

**Interjections:** Both protocol for the orthographic transcription contain a list of frequently used interjections (e.g. *uh*, *hè*). In addition, the CGN protocol has the option to mark interjections that do not appear on the list with *t.

### Complementary spelling guidelines - CGN

**Use of capital letters:** Proper names, such as cities, persons, brands and companies, start with a capital letter. When a proper name consists of more words, each word starts with a capital, even when this is not according to the standard spelling rules (e.g. *Anne Marie Van De Zande*). For titles of books, songs, films, etc., the same rules apply as for proper names.

**Pronunciation:** All words that are not contained in the lexicon of the CGN and also do not belong to any of the other types are marked with *u. Within these category three kinds of words can be distinguished. The first group are the onomatopoeic words (e.g. *boink*u*). The second group contains the words that are pronounced wrongly, either by accident or on purpose (e.g. *toekenbas*u* instead of *boekentas* ("book bag"), *alduns*u* instead of *aldus* ("thus")). The third group are the mispronunciations and resumptions within words (e.g. *gewee-weest*u* for *geweest* ("been"), *ver-uh-kocht*u* for *verkocht* ("sold")).

**Complementary spelling guidelines - CHILDES**

**Phrasal combinations:** In phrasal combinations of different word classes are combined. These include book titles (e.g. *Wuthering Heights*), names of places (e.g. *University of Oxford*), and lines from songs (*With a little help from my friends*). To indicate that these words form a phrasal combination the underscore character is used (*Wuthering_Heights*, *University_of_Oxford* and *With_a_little_help_from_my_friends*).

**Unidentifiable material**

**Unintelligible speech:** Words or phrases that are difficult to understand are marked with *x in the CGN. When they are completely unintelligible, the transcriber uses xxx instead of the word or phrase. This corresponds with the way CHILDES deals with unintelligible speech in which this is represented with "`xxx`" or "`xx`". The string "`xxx`" will be ignored when computing the mean length of utterances and other counts. The string "`xx`" will be counted as one word. To indicate that a transcribed word or phrase is a best guess the word or phrase is followed by "`[?]`" (e.g. *I want a frog [?].*, transcriber is not sure of the word *frog*).

**Non-speech acoustic events:** CHILDES and the CGN both provide rules for transcribing non-speech acoustic events. For the CGN clearly audible speaker sounds, such as laughter, crying, screaming or coughing, are represented by ggg (when relevant for the conversation) whereas in CHILDES this is transcribed by "`0`".

**Phonological fragments:** The CGN protocol provides the characters *a to mark phonological fragments (e.g. *ik ga mo*a nee overmorgen naar de tandarts.* ("to-m*a no the day after to-morrow I'm going to the dentist.")). When a complete word is repeated, the word is not marked (*wat wat deed je daar dan?* ("what what were you doing there?")). In CHILDES, phonological fragments are proceeded by "&" (e.g. *&t &t &k can't you go?*).

**The nonfluent speech**

The orthographic transcription protocol of the CGN has been used for transcribing the aphasic speech. However, although the transparency criterion is very important, some typical problems frequently present in aphasic speech ask for additional rules. These problematic phenomena - the interjections problem, the word finding problem, the produced versus intended utterance problem, the boundaries problem and the gestures problem - are discussed in more detail.

**Interjections**

Nonfluent aphasic patients need much time to think and utter many interjections (most times uh and uhm). According to the CGN guidelines, all interjections have to be transcribed:

**Example 6.4.1 (Interjections - 1a).** ***uh uh*** *de bed helemaal* ***uh uh*** *vliegen* ***uh*** *nou zes* *stoelen en* ***uh*** *en een gordijntje d'r omheen* ***uh*** *.*

(***uh uh*** *the bed all* ***uh uh*** *fly* ***uh*** *well six chairs and* ***uh*** *and a curtain around it* ***uh*** .)

Although the interjections may not seem very informative at first sight, they can give an indication of the efforts it costs to produce speech. Therefore, leaving them out of the transcription is not a good option. The transcription of the sentence then becomes:

**Example 6.4.2 (Interjections - 1b).** *de bed helemaal vliegen nou zes stoelen en en een gordijntje d'r omheen .*

If this option is adopted, information about the conversation is lost. Readers of the transcription get a completely wrong view of the conversation: it seems that the aphasic patient has a fluent production. The conversation also becomes more difficult to interpret because interjections can also indicate a new attempt of the aphasic speaker to convey the message in another way.

We devised a third option to transcribe the interjections properly. First, we thought of counting the interjections and indicating in the transcription how many interjections were uttered. However, whether this would be a good way to measure speaking effort, is doubtful. A speaker can say "uh, uh, uh" a number of times in succession, but it is also possible that a speaker says "uhhhhhhhhhhh". In this case one "uh" can last as long as five or six "uh"Šs. To measure the effort, it is more relevant to know the time employed by the speaker to produce the relevant utterance. So, the best solution would be to indicate filled pauses (<fp>) and to link the transcriptions to the recordings, in order to include information on the timespan (this is also done in the CGN). The orthographic transcription then becomes:

**Example 6.4.3 (Interjections - 1c).** *<fp> de bed helemaal <fp> vliegen <fp> nou zes stoelen en <fp> en een gordijntje d'r om heen <fp> .*

Adopting this option makes it easier to perform the orthographic transcription and little information is lost.

### Word finding problems

By definition, all nonfluent aphasic patients experience word finding problems. While searching for the right word, they may produce several other related words. We believe it is relevant to mark words and phrases uttered during the word finding process since in this way we will increase the readability and make it possible to filter out these words. It will also be possible to find out which word categories typically cause word finding problems.

The patients involved in the pilot study encountered difficulties in finding numerals, geographical locations, and time indicators. In the example below, the patient searches for the country *Frankrijk* ('France').

```
<i> ok en waar was je precies ? </i>
```

```
<p> Valdorand . </p>
<i> en waar ligt dat ? </i>
<p> uh in Zwitserland niet maar uh uh Valdorand uh Duitsland
    Oostenrijk Zwitserland uh Oostenrijk Oostenrijk Zwitserland .
    </p>
<i> nee hij is even weg ? </i>
<p> nee uh Duitsland uh Zwitser*a he Valdorand uh . </p>
<i> in het buitenland . </i>
<p> ja uh Oostenrijk niet Zwitserland niet Spanje niet . </p>
<i> je hebt ze allemaal voor je de landen maar . </i>
<i> hij is even weg ? </i>
<i> nou misschien dat je d'r zo op komt . </i>
<p> ja . </p>
```

In the orthographic transcription according to the CGN guidelines, it is not possible to indicate that all countries (*Zwitserland* ('Switzerland'), *Oostenrijk* ('Austria'), *Duitsland* ('Germany') and *Spanje* ('Spain')) are produced during the word finding process of the country *Frankrijk*. In one of the CHILDES corpora, the Holland Corpus, this is encoded by putting the words that are uttered during the word finding process between angle brackets. This makes it possible to filter out only the relevant words. Another way of indicating that a word was produced during the word finding process is to mark it with *wf followed by the intended word. The orthographic transcription of the relevant part of the example would then be:

```
<p> uh in Zwitserland*wf(Frankrijk) niet maar uh uh Valdorand uh
    Duitsland*wf(Frankrijk) Oostenrijk*wf(Frankrijk)
    Zwitserland*wf(Frankrijk) uh Oostenrijk*wf(Frankrijk)
    Oostenrijk*wf(Frankrijk) Zwitserland*wf(Frankrijk) . </p>
<i> nee hij is even weg ? </i>
<p> nee uh Duitsland*wf(Frankrijk) uh Zwitser*a he Valdorand uh .
    </p>
<i> in het buitenland . </i>
<p> ja uh Oostenrijk*wf(Frankrijk) niet Zwitserland*wf(Frankrijk)
    niet Spanje*wf(Frankrijk) niet . </p>
```

The DTD of CGN XML can be extended to make it possible to mark these words with a special markedness category, e.g. "word_finding". The word to be found can also obtain an attribute, e.g. "wordtobefound". When we would transcribe word finding difficulties in this way, the word "Zwitserland" in the example would be transcribed as:

```
<w id="fn..." marked="word_finding" wordtobefound="Frankrijk">
Zwitserland</w>
```

It is also possible that a word is not found at all. Words produced during the word finding process can be marked then with *wf, without the word to be found indicated between brackets thereafter.

**Produced utterance vs. intended utterance**
Produced words are sometimes (slightly) different from the intended words, it is clear what the speaker wants to say, but the realization of the word is not completely correct (e.g. *legepodie* instead of *logopedie* (speech therapy)). Such errors are marked with `*u`, the marking used in the orthographic transcription of the CGN to indicate that a word is a mispronunciation (either by accident or on purpose) or an onomatopoeic word. Although the errors of the aphasic speakers are not exactly the same as the mispronunciations produced by speakers with intact speech abilities, this is the category that comes most close. It would be better if such errors could be marked in a special way, for example with `*i`.

**Distinguishing utterances**
Nonfluent aphasic patients speak in short, often ungrammatical phrases with many pauses. They generally leave out function words and word order is disturbed. It is very difficult to specify utterance boundaries since sentences are often not completed or finished after another sentence has been produced. It would help the transcriber if guidelines to detect the boundaries are given.

Although distinguishing utterances will always remain a subjective issue, it is possible to define some guidelines that can be used to decide where a new utterance starts. One possibility is to look for a topic shift. When this would be the case, it could be a clue to start a new utterance. Topics often contain more then one utterance, so it is still possible to miss boundaries in this way. Another option is to look for pauses. When a long pause is 'heard', this could be a clue for starting a new utterance. However, while this might be a good clue in speech from persons without speech disabilities, this is not always the case in aphasic speech. Pauses are very common in this kind of speech, since they are also used within utterances. Even in normal speech a pause does not always mark a boundary. A third clue could be the intonation pattern (Wijckmans and Zwaga 2005): a decreasing intonation pattern indicates an utterance boundary. However, intonation might be disturbed for some aphasic patients, sometimes they speak in a rather monotonous tone.

**Gestures**
For the encoding of gestures, the Holland corpus gives a possible solution. The non-speech acoustic events that influence the conversation are clearly encoded in the Holland transcripts. In the Holland corpus, fragments of non-speech acoustic events are coded by `[% non-speech acoustic event]`, e.g. `[% laugh]` for encoding laughing. Not only actions as laughing are transcribed, but also all other non-speech events, such as taking something. Unfortunately, the Holland corpus has not been converted to XML. In XML, a solution could be to use a seperate tier parallel to the speech to annotate the gestures. For the annotation of gestures, it would be of much help if it would be allowed to make video recordings of the conversations.

### 6.4.4 Part-of-Speech Tagging

For the part-of-speech tagging, the approach of the CGN was adopted. The results of the automatically performed tagging of the aphasic speech where compared to the results obtained within the CGN project. Because of the size of the CGN, part-of-speech tagging was automated as much as possible. The TiMBL (Tilburg Memory-Based Learner) combitagger was used (Daelemans, Zavrel, van der Sloot and van den Bosch 2004). This tagger systematically compares the results of four separate working taggers in order to obtain a result that is more accurate then the results the individual taggers can give. The result of the automatic tagging and lemmatization has been verified and corrected manually. The performance of the combitagger on the CGN after retraining was 96.6% (Oostdijk et al. 2002).

#### Nonfluent speech

Aphasic nonfluent speech differs from spontaneous speech by persons with intact speech abilities. To investigate how automatic part-of-speech taggers actually perform on nonfluent speech, a subset of the automatically tagged data has been checked manually. The used tagger is one of the four taggers that was incorporated into the combitagger that has been used for the annotation of the CGN, namely the Memory-Based Tagger (MBT) (Daelemans and van den Bosch 1996)

MBT uses a memory-based learning approach to tagging. In this approach, a set of example cases is kept in memory. Each example case consists of a focus word with preceding and following context (two positions to the left and two positions to the right) and the category for that word in that specific context. New sentences are tagged by mapping each word to the most similar example case. For the construction of a POS-tagger for a specific corpus, an annotated corpus is needed. From this annotated corpus three data structures are extracted: a lexicon, a case base for known words, and a case base for unknown words. During tagging of new text, each word is looked up in the lexicon. When a word is found, it is disambiguated using the context to decide what the most similar case is. When a word is not contained in the lexicon, the tag for that word is based on its form, its context, and the most similar cases in the lexicon. The output is a best guess of the category for the word in its current context.

After tagging with MBT, one third of the data has been verified manually. For each word, it is indicated whether it is spoken by the aphasic patient or by the interviewer. All tagged words are classified as correct, wrong, interjection or punctuation mark. The interjections and punctuation marks have been separated from normal words because for the aphasic patients 36.6% of the words consists of interjections and punctuation marks, whereas for the interviewer this is only 19.7%. The interjections - as far as they are recognized by the tagger - and punctuation marks are always tagged correct. In the comparison of the utterances of the two groups (patients and interviewer), they are left out in order to prevent that the results are influenced by the large number of interjections used. The percentage of words that are assigned a wrong tag is 21.3% (183/860) for the patients whereas this percentage for the interviewer is only 15.8% (90/570). This difference is significant, $X^2(1, N = 1430) = 6.688$, $p \leq 0.05$, so the tagger performs better on the utterances of the interviewer.

|  | Correctness |  |  |
|---|---|---|---|
| Subject | Not correct | Correct | Total |
| Interviewer | 90 (15.8%) | 480 (84.2%) | 570 |
| Patients | 183 (21.3%) | 677 (78.7%) | 860 |
| Total | 273 (19.1%) | 1157 (80.9%) | 1430 |

TABLE 5 Tagger correctness for interviewer and patients

Further evaluation of the data showed that the errors can be divided roughly in five categories. The main error categories differed for the two kinds of speech. Within these categories, subcategories can be distinguished. The most occurring problem in the interviewer's speech was tagging the pronoun *je* ('you', 44.4% within error category "Same POS-tag"). The problem with *je* was that the tagger often tagged it as an indefinite pronoun instead of a personal pronoun. For the speech of the aphasic speakers the most problematic in the within error category was the tagging of capital letters (71.9% of all errors).

The three main reasons for assigning a wrong tag in the aphasic speech were:

- Words marked with a * in the orthographic transcription (29.5%)
- Unknown interjections, most times *uhm* or *ok* (11.5%)
- Capitals, e.g. N, A, D (14.2%)

For the speech produced by the interviewer the main problems were:

- Unknown interjections, most times *uhm* or *ok* (34.4%)
- Tagging the pronoun *je* as an indefinite pronoun instead of a personal pronoun (13.3%)

All other errors did not occur frequently and involved, among others, words with diacritic marks (e.g. *één* ('one')) and ambiguous words (e.g. *vier*, which means either "four" or "celebrate").

## Improving the performance of the Memory-Based Tagger

There are several ways to improve the performance of MBT on the speech of both the aphasic patients and the interviewer. The performance of MBT heavily depends on the quality of the training corpus. Therefore, the best way to improve the over-all performance accuracy, is to base the tagger on a manually tagged training corpus of the target speech, in this case on speech produced by aphasic patients. This will probably result in a lower error rate, mainly in the common error categories, such as the tagging of capitals. The problem of unknown interjections can be solved by adding them to the vocabulary of interjections. Words marked with an * should be excluded from the tagging process and get no tag at all. Dealing with abbreviated words, such as *da's* (that is) and *'t* (it), should be improved. The abbreviations consisting of two words (e.g. *da's*) should be separated during the tokenization process and tagged as two words. Abbreviations of one word (e.g. *'t*) should be learned from the training corpus. However, for tagging the CGN these abbreviations were not problematic, so maybe our bad results on this point are due to using only one of the taggers of the combitagger. Finally, the tagger should be able to deal with words with diacritic marks.

## 6.5 Conclusions

The pilot study we have carried out is a preliminary investigation for the setup of a Corpus of Dutch Aphasic Speech. Corpus design issues have been examined and we have especially focused on whether existing annotation and transcription protocols such as those developed within the CGN project or CHILDES could be employed within CoDAS.

We can conclude that the orthographic transcription protocol of the CGN is not completely suited for aphasic speech and special attention has been dedicated to features that are typical of this kind of speech such as interjections, word finding difficulties and the problem of distinguishing utterances.

The performance of MBT, one of the four automatic part-of-speech taggers used for the tagging of the CGN, on the tagging of the orthographic transcriptions of the Dutch aphasic speech, was worse than the performance of the combitagger on CGN annotation. Some main error categories can be distinguished. Training MBT on a corpus of manually tagged aphasic speech will probably result in a better performance of the tagger. Especially the type of errors contained in the main error categories will cause less problems if the tagger is trained on aphasic speech.

Besides orthographic transcription and part-of-speech tagging, we also investigated in the pilot study whether the phonetic transcription procedure of the CGN could be adopted. For a small part of the data, the automatically generated transcriptions have been checked globally. At first sight there seemed to be few problems. A more detailed investigation of the results is needed to draw strong conclusions (Westerhout 2006). The investigation of the problems that aphasic speech constitute for syntactic and prosodic annotation is left for future research.

## References

Binnenpoorte, D.(2006), *Phonetic Transcriptions of Large Speech Corpora*, PhD thesis, Radboud University.

Cucchiarini, C., van Hamme, H., van Herwijnen, H. and Smits, F.(2006), Jasmin-cgn: Extension of the spoken dutch corpus with speech of elderly people, children and non-natives in the human-machine interaction modality, *Proc. 5th International*

*Conference on Language Resources and Evaluation*, Genoa, Italy, pp. 135–138.

Daelemans, W. and van den Bosch, A.(1996), Language-independent Data-oriented Grapheme-to-phoneme Conversion, *in* J. Van Santen, R. Sproat, J. Olive and J. Hirschberg (eds), *Progress in Speech Synthesis*, Springer Verlag, pp. 77–90.

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A.(2004), TiMBL: Tilburg Memory-Based Learner, *Technical report*, Induction of Linguistic Knowledge (ILK), Tilburg University.

Davidse, W. and Mackenbach, J.(1984), Aphasia in the Netherlands; Extent of the Problem, *Tijdschrift voor Gerontologie en Geriatrie* **15**(3), 99–104.

Goedertier, W., Goddijn, S. and Martens, J.(2000), Orthographic transcription of the Spoken Dutch Corpus, *in* M. Gravilidou, G. Carayannis, S. Markantonatou, S. Piperidis and G. Stainhaouer (eds), *Proceedings of LREC 2000*, Vol. II, pp. 909–914.

MacWhinney, B.(2000), *Transcription Format and Programs*, Vol. 1 of *The CHILDES project: tools for analyzing talk*, Lawrence Erlbaum.

Oostdijk, N., Goedertier, W., van Eynde, F., Bovens, L., Martens, J., Moortgat, M. and Baayen, H.(2002), Experiences from the Spoken Dutch Corpus Project, *in* M. Gonzalez Rodriguez and C. Paz Saurez Araujo (eds), *Proceedings of LREC-2002*, pp. 340–347.

Westerhout, E.(2006), *A corpus of dutch aphasic speech: Sketching the design and performing a pilot study*, Master's thesis, Department of Linguistics, Utrecht University, Utrecht, The Netherlands.

Westerhout, E. and Monachesi, P.(2006), A pilot study for a Corpus of Dutch Aphasic Speech (CoDAS), *Proc. 5th International Conference on Language Resources and Evaluation*, Genoa, Italy, pp. 1648–1653.

Wijckmans, E. and Zwaga, M.(2005), ASTA: Analyse voor Spontane Taal bij Afasie. Standaard volgens de VKL.

Zavrel, J. and Daelemans, W.(1999), Evaluatie van Part-of-Speech taggers voor het Corpus Gesproken Nederlands, *Rapport CGN: werkgroep corpusannotatie*, Tilburg University.

7

# Where FrameNet meets the Spoken Dutch Corpus: in the middle

PAOLA MONACHESI AND JANTINE TRAPMAN

*Utrecht University, Uil-OTS*
*Trans 10, 3512 JK Utrecht, The Netherlands*
*Paola.Monachesi@let.uu.nl, Jantine.Trapman@let.uu.nl*

## Abstract

In this paper, we investigate to which extent FrameNet could be employed to enrich a syntactically annotated corpus such as the Corpus of Spoken Dutch with semantic role information. To this end, we have taken a language specific phenomenon such as the Dutch adjunct middle construction, as a test case.

## 7.1 Introduction

The interest for semantic annotation of corpora has grown in the last years. Applications such as information extraction, question-answering, document classification, and automatic abstracting that are based on underlying probabilistic techniques benefit from large corpora for improving their results and this is especially the case if these corpora are enriched with semantic information.

Several initiatives have been launched at the international level showing that it is pos-

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

sible to obtain concrete results with respect to the annotation of corpora with semantic information. Projects such as PropBank (Kingsbury, Palmer, and Marcus 2002), which has focussed on annotation of argumentstructure, have demonstrated that creating semantically annotated corpora need not be extremely expensive, and that it is possible to achieve a remarkable degree of consensus on a theory-neutral annotation methodology. On the other hand, projects such as Framenet (Johnson et al. 2002) have shown that it is possible to reach a considerable degree of granularity in the encoding of semantic roles.

However, while most initiatives have focused on English, not much attention has been dedicated to the creation of semantically annotated Dutch corpora, notably the Corpus of Spoken Dutch (CGN) lacks a layer of semantic annotation (Oostdijk et al. 2002). There is the need for appropriate guidelines with respect to the semantic annotation of Dutch corpora which could be adopted both for the annotation of the written Dutch corpus developed within the D-coi project (http://lands.let.ru.nl/projects/d-coi/) and for the already existing CGN.

In this paper, we discuss one type of semantic annotation, that is semantic role assignment. Semantic roles express the relationships identified between items in a text, such as the agents or patients of particular actions. The reason for our choice to focus on role assignment lies in the fact it is a thoroughly attested and feasible type of semantic annotation within corpora such as the already mentioned Framenet and PropBank projects and SALSA, (Erk et al. 2003) which takes the FrameNet dictionary as its basis.

We base our investigation on the already existing CGN in order to establish whether the annotation of semantic roles proposed within the FrameNet project could be adopted for Dutch and to which extent it can be integrated with the syntactic layer already present in CGN. The results, however, should be applicable also to a written corpus such as the one developed within the D-coi project.

Within the FrameNet project, a frame semantic lexicon has been developed which tries to encode all possible semantic and syntactic contexts for each entry. Moreover, the underlying frame ontology makes it possible to relate entries not only through membership of the same frame but also by means of inheritance relations. FrameNet is still under development, however, its methodology has been adopted to develop FrameNets for languages other than English. One important initiative in this respect is the German project SALSA, (Erk et al. 2003) which is not restricted to building a lexicon but it annotates the complete German Tiger corpus, (Brants et al. 2002) using the FrameNet dictionary and adapting it to German.

In order to assess whether the FrameNet lexicon can be employed to annotate a Dutch corpus with semantic role information, we have taken a specific phenomenon into consideration: the adjunct middle construction. This construction is quite similar to the object middle, which occurs both in English and Dutch. However, the adjunct middle does not occur in English (Hoekstra and Roberts 1993) and therefore it seems an appropriate test case to verify whether FrameNet can be adopted and eventually extended to deal with a language specific phenomenon. The adjunct middle construction constitutes a relevant phenomenon also because it is characterized by specific syntactic constraints as well as certain peculiar semantic properties which makes it a relevant case study for the interaction between syntactic and semantic annotation in corpora.

In the next section, we provide a detailed description of the various properties of the adjunct middle construction in Dutch, while in section 7.3 a brief introduction to the FrameNet project is given. Section 7.4 shows how the various adjunct middle verbs can be classified according to FrameNet frames while in section 7.5 the semantic roles which are involved in this construction are presented. Finally, section 7.6 discusses how the FrameNet lexicon can be employed to annotate the Corpus of Spoken Dutch, while 7.7 contains some concluding remarks.

## 7.2 The adjunct middle

The middle construction is characterized by an active voice, in the form of an intransitive verb, or a transitive verb that is used intransitively. Furthermore, a non-Agent is promoted to the subject position. An example is given by the active sentence in (12a) which can be transformed into the middle sentence in (12b). While sentence (12a) contains an Agent in the subject position and a Theme in the position of the direct object, in (12b) the Agent is no longer syntactically present and the direct object is now in the position of the subject:

(12) a. De padvinder schilt de aardappelen met een mesje.
The boy scout peels the potatoes with a knife
'The boy scout peels the potatoes with a knife'

b. Deze aardappelen schillen makkelijk.
These potatoes peel easy
'These potatoes peel easily.'

This type of construction, the object middle, is attested both in English and in Dutch, but in Dutch, another type of middle construction can be employed: the adjunct middle. It is characterized by the presence of an adjunct in the subject position, as exemplified by example (13a) below. No object is present in the middle construction which is consistent with its purpose: to focus on the (former) adjunct. In addition to adjuncts, demonstratives and the particle *het* ('it') can also occur as subjects, as shown in (13b), eventually in combination with *zijn* ('be') and an infinitive verb, as exemplified in (13c):[27]

(13) a. Dit mesje schilt handig.
This knife peels neat
'This knife is neat for pealing.'

b. Dat/het fietst prettig hier.
That/it cycles nice here

[27]The examples in this section are taken from (Ackema and Schoorlemmer 1993), (Ackema and Schoorlemmer 1995), (Haeseryn et al. 1997), (Peeters 1999) and (Hoekstra and Roberts 1993).

'It is nice to cycle here.'

c. Het is hier lekker zitten.
It is here nice sitting
'It is nice to sit here.'

The middle owes its name to the fact that it shares some of its properties with passives on the one hand, while on the other hand it shows some similarities with ergatives. In the rest of this section, the most important properties of the Dutch adjunct middle construction are summarized. Special attention is dedicated to those characteristics which directly affect the syntactic structure or the interpretation of the relationship between the verb and its arguments. These properties will eventually enable us to:

- identify the adjunct middle construction within the syntactically annotated data of the CGN;
- to assess whether we can represent it correctly within the theory of Frame Semantics as exemplified in FrameNet.

In particular, we will discuss the type of verbs which can be attested in this construction, the constraints on the subject, the presence of an implicit Agent as well as that of the compulsory modifiers, for more details we refer to (Peeters 1999).

**The adjunct middle verb** Not all verbs allow middle formation. The ones which allow adjunct middle formation are mostly intransitives although there is a number of verbs which allow both object and middle formation. However, If a verb of the latter group appears in a middle construction its object cannot be present. (Peeters 1999) divides the intransitives that trigger middle formation into three classes:

1. verbs of position;
2. verbs of physical activity, implying no locomotion;
3. (agentive) verbs of manner of motion (expressing no directional endpoint).

**The subject** The grammatical subject in an adjunct middle construction has to meet certain syntactic and semantic requirements. Three types of adjuncts are allowed in the subject position, that is an instrument (14a), a location (14b) or an external circumstance (14c), as shown by the examples below:

(14) a. Deze stoel zit lekker.
This chair sits comfortable
'This chair is comfortable to sit on.'

b. Deze sportzaal turnt prettig.
This gym does gymnastics nicely
'In this gym it is nice to do gymnastics.'

c. Regenweer wandelt niet gezellig.
Rainy weather walks not pleasant

'It is not pleasant to walk in rainy weather.'

In a regular matrix clause, these adjuncts are preceded by a preposition, but in the middle construction these prepositions have disappeared, as a comparison between (15) and (14a) reveals:

(15) Men zit lekker op deze stoel.
One sits comfortable on this chair
'One sits comfortably on this chair.'

(14a) Deze stoel zit lekker.
This chair sits comfortable
'This chair is comfortable to sit on.'

It is the following hierarchy which regulates the degree of acceptability of adjuncts:

$$\text{Instrument} \ll \text{Location} \ll \text{External Circumstance}$$

The leftmost element is the most eligible for middle formation while elements more to the right are less eligible. Thus, a middle verb which allows an adjunct of external circumstance in the subject position, automatically allows a Location or an Instrument in that position. As we have mentioned before the focus of the adjunct middle is on its subject which makes the presence of another element (e.g. an object, a purpose clause) not desirable. An additional constraint is that the subject should not represent a human entity.

**The Agent** The prototypical adjunct middle construction contains an Agent which does not surface in syntax, but is only implicitly present at the semantic level. The Agent can be characterized by the features [+animacy] and [+volitionality] (i.e. conscious and deliberate), but it is often interpreted as [+human]. In the agentive counterpart of the middle construction, the Agent is indicated by the arbitrary (pro)noun *men* ('one', 'people'), as illustrated by example (16a) compared to (16b), which represents the adjunct middle version of (16a):

(16) a. Men loopt lekker op deze schoenen.
One walks nice on these shoes
'One walks nicely on this shoes.'

b. Deze schoenen lopen lekker.
These shoes walk nice
'On these shoes one walks nicely.'

Only under certain conditions, it is possible for an Agent to appear explicitly in the middle construction. In this case, it is represented by a PP introduced by the the preposition *voor* ('for'), this is possible in the case the Agent is generic or non-specific, as shown in (17a):

(17) a. Een krukje zit vervelend voor oude mannen /een oude man /?Hans.
A stool sits tedious for elderly men /an old man /?Hans
'A stool is tedious to sit on for elderly men / an old man /?Hans.'

b. Dit ijs schaatst goed genoeg voor Hans.
This ice skates good enough for Hans
'For Hans this ice is good enough to skate on.'

A sentence like (17b), where the Agent is a referential expression, is only allowed if the modifier has a restrictive, hence a comparative meaning.

**The modifier** The modifier encodes information on how the action of the predicate can be carried out with respect to the entity specified by the subject (Fagan 1992). The modifying element can be an adjective, as shown in the previous examples, negation (18b) or a stressed element (18a) and it has a dyadic character; On the one hand, the modifier refers to the subject, on the other hand, it is needed to identify the Agent:

(18) a. Dit ijs SCHAATST.
This ice skates
'This ice DOES skate.'

b. Dit ijs schaatst niet / lekker / *glad.
This ice skates not / nice / smooth
'This ice does not skate / skates nicely / *skates smoothly.'

Modifiers which are exclusively related to the subject or the Agent are excluded from middle formation, as is the case for the adverb *glad* 'smoothly', in example (18b).

Due to the presence of the modifier, an implicit division automatically arises among the set of elements to which a certain property does (not) apply. This division can be quite explicit, as in (18b), where the distinction is made between ice that does skate (nicely) and ice that does not skate (nicely).

**The semantics of the adjunct middle** The adjunct middle construction focuses on (the properties of) the instrument, location or external circumstance, instead of the Agent. The passive sentence shows a similar character: the direct object occupies the position of the subject. Although the middle has some properties of passives, it is not sufficient to assign it a passive meaning as (Fagan 1992) does: "being able to be V-ed." (Peeters 1999) gives a somewhat different meaning description for the middle with structure 'NP V X': "Adjunct NP enables whomever, to (un)succesfully V." The role of the modifier is left out of both descriptions, but could be filled in by adding "in an X manner".

Furthermore, the adjunct middle has the following semantic characteristics:

- non-eventiveness;
- it does not express or imply a completed change of location or state;

- it does imply an Agent;
- the Agent does not control the quality of the process (the Agent is more like an Experiencer);
- the modifier provides the middle with a comparative character.

After this general introduction of the properties of the adjunct middle, we will discuss in the next sections whether FrameNet can be assumed to classify the adjunct middle verbs according to its frames and whether the various elements of this construction can be labelled with appropriate semantic roles labels.

## 7.3 FrameNet

The Berkeley FrameNet project is based on the theory of Frame Semantics. Each frame represents a system of concepts related to each other (Petruck 1996). Words derive their meaning from the frame they belong to and their meaning is related to other words.

An example to illustrate the way FrameNet is structured can be given on the basis of the concept *buy*. The concept *buy* is included within a more abstract frame containing related concepts, e.g. *rent, spend, pay, cost* in this case. In FrameNet, this frame is called Commerce_buy (Johnson and Fillmore 2000). Concepts within the same frame may differ from each other due to the way in which the action is carried out, for example: pay with a bank/chip card or pay cash or because of the person involved in the transaction as in the case of *buy* vs. *sell*.

Besides the concepts which can be evoked in a frame, that is the so-called Frame Evoking Elements (FEEs) there are also Frame Elements (FEs) present in a frame. The elements *Buyer, Goods, Seller, Money* belong to the core of the concept associated with the verb *buy*. These frame elements represent the situational roles of the predicate. In case of *buy* the *Buyer* and *Goods* are obligatory, the other roles are optional. This information is encoded in the typical scenario which is described by a definition that covers all the possible contexts: each concept has such a prototypical scenario as basis.

The frame comprises a frame definition, a list of frame elements and a list of lexical units – the frame evoking elements. A lexical unit (LU) spells out all the various meanings of a word. The lexical entry encodes the valence description showing, by means of illustrative sentences, the various semantic and syntactic structures in which a LU can appear together with its frame elements. If a word has four different meanings, it has four lexical entries in FrameNet.

The complete description of a verb thus contains its frame definition, the elements of that frame, the grammatical properties of the verb and the various syntactic patterns in which it can appear (Petruck 1996). One problem concerning frame labels is that several parts of a sentence can evoke several frames simultaneously.

Not only lexical units are related to one another, frames themselves are mutually connected as well by means of subframes and *inheritance* or *using* relations. Inheritance is a

| De stoel | zit | lekker | (Frame: Posture) |
|---|---|---|---|
| Location | | Depictive | CNI: Protagonist |
| Ext | | Mod | |
| NP | | AdvP | |

FIGURE 10 An adjunct middle sentence in FrameNet

"IS-A"-relation between the mother frame and a daughter. The daughter inherits the semantic (sub)type and the subframe structure from the mother. In addition, a daughter can include extra frame elements. The difference between *inheritance* and *using* is that the former implies complete inheritance whereas the latter involves incomplete inheritance. Notice that a daughter can have several mothers.

In summary, FrameNet is built out of three components (Fillmore, Baker, and Sato 2004):

1. the frame ontology (the set of frames)
2. the set of annotated sentences (examples of evoking the frames)
3. the set of lexical entries

The example in figure 10 illustrates how an adjunct middle sentence can be represented using FrameNet. The annotation of FrameNet encodes not only information about FEEs and FEs but also information about the syntactic function of the elements involved and information about their part of speech. The verb from our example evokes the frame **Posture** which is associated with the following definition: " The words in this frame describe the stable body posture of an Agent". *Protagonist, Depictive, Direction, Distance, Goal (e.g. lean against the wall), Location* and *Manner* are some FEs related to this frame. The adjective *lekker* constitutes also an FEE; it evokes the frame **Aesthetics**. But since this paper is only concerned with argument structure, the adjectival FEE is not discussed further. The Agent, which is called here the Protagonist, is syntactically absent, but it is present at the semantic level. In FrameNet, it is expressed at the end of the clause it belongs to, and the tag CNI: Constructionally licensed Null Instantiation is used to express this information.

## 7.4 Classifying adjunct middle verbs according to FrameNet frames

After this brief overview of the FrameNet system, we can now assess whether it can be employed to annotate the Dutch adjunct middle construction. The first step in this process is to establish to which frame a given verb belongs: the existing frame classification of FrameNet is used for this purpose.[28] The classification is based on English, but our assumption is that it should also be applicable to Dutch. In the rest of this section, we discuss under which frames the Dutch middle verbs can be grouped and which similarities and relationships these frames share.

[28]A complete overview can be found on the FrameNet website: http://framenet.icsi.berkeley.edu/

We have investigated sixty verbs which are extracted from example sentences in the literature and classified according to the three categories proposed by (Peeters 1999). They are listed in figure 11.

Each verb evokes one or several frames and different verbs can of course evoke the same frame. Since FrameNet is still under developement, it is incomplete; it does not contain every middle verb from our list. In those cases where the verb was not found in FrameNet, we have tried to assign it to an existing frame or to introduce a new frame if there was no appropriate one available.

A list was made of the frames that contain one or more middle verbs and if there were also non-middle verbs in the frame, we have verified whether they were eligible for middle formation. Finally, we have investigated how the frames that contain middle verbs are related to each other. This could be a direct relationship in which one frame inherits from or uses another frame. However, the relation could also be more indirect in the case two or more frames have the same mother.

For example, all the verbs belonging to the first colum, in figure 11, that is verbs of position evoke the frame **Posture**. All the additional verbs belonging to this frame can undergo middle formation in Dutch.

The other verbs listed in figure 11 belong to the frames summarized in figure 12, in which the various relations among frames are illustrated. Our aim was to generalize over types of verbs and frames which can be evoked in the adjunct middle construction. Figure 12 shows that middle verbs cannot be grouped under one frame but they belong to several ones. The most important mother frames are **Posture** (previously discussed) as well as **Intentionally_act** and **Motion**, represented in figure 12, which, however, are not connected with each other. The second frame itself does not contain middle verbs but it is included in the diagram because it has several daughters that do. It should be noticed that frames containing only one of the sixty investigated verbs include other verbs that can undergo middle formation, but also many verbs that cannot. Hence, we cannot simply state that if one frame includes some middle verbs, all the other verbs belonging to this frame can undergo middle formation. Furthermore, we should point out the presence of the **Sport** frame in figure 12. This frame does not exist in FrameNet, however, we have introduced it to group middle verbs that express sporting activities. The relations between the **Sport** frame and the other frames are only generally sketched. It should be left to the developers of FrameNet to assess the validity of this introduction further.

## 7.5 Assigning Frame Elements to adjunct middle verbs

In order to provide a complete representation of adjunct middle sentences, it is necessary to assign a label to the adjunct which is in the subject position, to the implicit Agent and to the modifier. Therefore, for each verb we checked which frame elements from the frame they evoked provided the suitable label. The list of frame elements (i.e. semantic roles) is ordered according to coreness and alphabetical order. So it seems that once we have

| **Verbs of position** | **Verbs of physical activity** | **Verbs of manner of motion** |
|---|---|---|
| hangen | breien | draven |
| leunen | dansen | fietsen |
| liggen | eten | galopperen |
| rusten | golfen | glijden |
| staan | gooien | klimmen |
| steunen | kaarten | lopen |
| zitten | koken | rennen |
| | laden | rijden |
| | lezen | reizen |
| | praten | schaatsen |
| | roken | skiën |
| | schaken | springen |
| | schermen | stappen |
| | schillen | varen |
| | schoonmaken | vallen |
| | schrijven | vliegen |
| | schudden | wandelen |
| | slapen | zeilen |
| | spelen (toneel) | zwemmen |
| | spelen (sport, spel) | |
| | tekenen | |
| | tennissen | |
| | turnen | |
| | typen | |
| | vechten | |
| | vegen | |
| | voetballen | |
| | vrijen | |
| | werken | |
| | winkelen | |
| | zingen | |

FIGURE 11 List of Dutch adjunct middle verbs analyzed

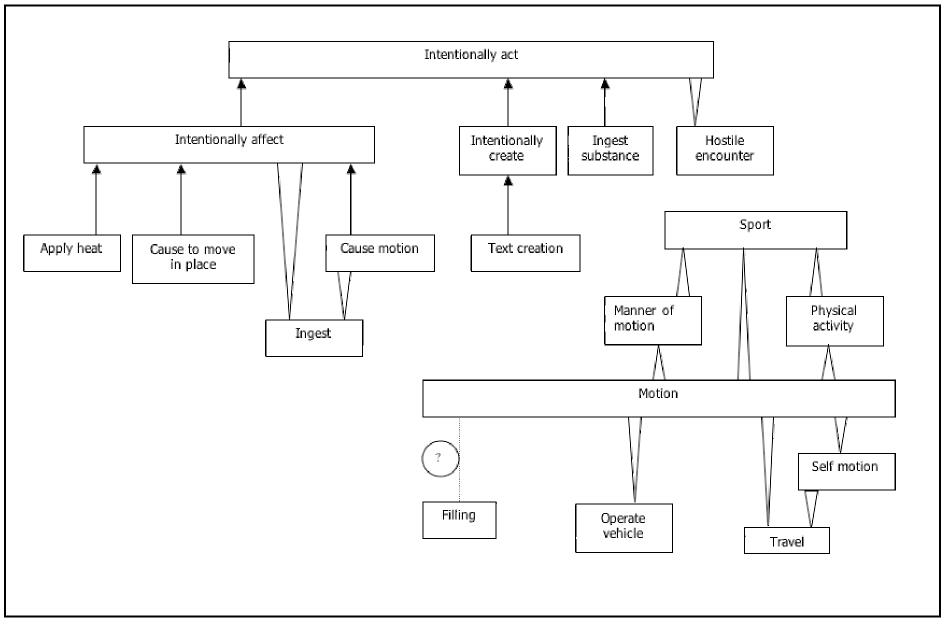


FIGURE 12 Frames including adjunct middle verbs and their relations

manually established to which frame a given adjunct middle verb belongs to, we have to detect the relevant frame elements whose label can be assigned manually to the various situational roles of the predicate.

In particular, for each verb, we have established which frame element would represent the (implicit) Agent. FrameNet uses various labels for what is traditionally called the Agent: e.g. *Ingestor, Self_mover, Cook, Interlocutor_1/Interlocutors, Author, Sleeper, Driver* allowing for a high degree of granularity but making it rather difficult to eventually automatize the annotation process.

Similarly, when it comes to possible adjuncts which can be on the subject position, three types are identified by (Peeters 1999), that is Instrument, Location and External Circumstance. FrameNet however, exhibits a higher degree of granularity. Therefore, for each relevant frame, we have established which frame element is allowed in the subject position of an adjunct middle verb. They can be both core and non-core elements. Potential subjects are *Goal, Theme, Instrument, Place, Area, Supporting_Bodypart, Vehicle, Circumstance*. From our investigation, it appears that frame elements with the same name are attested in various frames, however, not always with the same definition in each one. Therefore, since we cannot be sure that the description of a frame element is consistent through the whole lexicon we are obliged to examine for each element whether its definition varies across different frames. This uncertainty, in addition to the high degree of granulairty, makes it also in this case difficult to make the annotation process automatic.

In addition to the more fine grained labelling of adjuncts, there is another difference in terminology between FrameNet and the information found in the literature. In the sentence *De stoel zit lekker* ('The chair sits comfortably'), Peeters classifies the subject as an *Instrument*, while according to the description of **Posture**, *de stoel* ('the chair') is labelled as *Place*.

It is not standard that the modifier is present in a frame, however, if attested, it is usually represented as *Depictive*. For further details with respect to the classification of adjunct middle verbs according to FrameNet frames and for the labelling of the various semantic roles involved, we refer to (Trapman 2005).

## 7.6 Annotating the Spoken Dutch Corpus with FrameNet

In the previous sections, we have shown that it is possible to classify Dutch adjunct middle verbs according to FrameNet frames and to establish the semantic roles (Frame Elements) related to the various elements present in this construction. In this section, we illustrate how this information can be employed to enrich an existing corpus such as the Spoken Dutch Corpus with a semantic annotation layer. In particular, we discuss how a sentence in which the adjunct middle construction is attested can be annotated on the basis of the FrameNet information.

The Spoken Dutch Corpus includes about 8.900.000 words from both Flemish and Dutch sources including spontaneous conversations, telephone dialogues, news bulletins, read aloud texts etc. All together roughly 800 hours of spoken material in modern Dutch have been collected. The transcribed material has been enriched with part-of-speech tagging while a smaller part of the corpus has been annotated with phonetic, prosodic and syntactic information.

In order to indentify the adjunct middle construction in the corpus, we have employed the syntactically annotated part as well as the lexicon of the CGN. The middle construction can be identified as a predicate-argumentstructure which lacks an object, and some kind of AP has to be present within the dependency structure. Unfortunately, in the CGN, information about dependency structures and subcategorization is available but in two separate modules of the query tool. Therefore, the subcategorization information is not available while one is searching in the syntactic annotated part.

Despite these shortcomings, we were able to identify adjunct middle sentences correctly. In the rest of this section, we will provide some examples of annotation taken from the CGN, however, we will assume that the required information about subcategorization is available within the syntactic annotated corpus.

The sentences in figures 13, 14 and 15 are examples taken from the corpus. The annotation starts from the verb, which is the frame evoking element. A verb can evoke several frames at a time; other sentence elements determine the exact frame. In addition to their part-of-speech and their syntactic labels, lexical verbs, adjectival and nominal phrases get a semantic label, as well. In the case of the verb, the label represents the frame

(19) ’nou een luchtbed slaapt op zich wel heel erg fijn.’
well an air-bed sleeps in itself indeed very much comfortably
’well, an airbed in itself does sleep very comfortably indeed.’

<fn000682.326>

| word | pos | syn | sem |
|---|---|---|---|
| nou | BW() | | |
| een<br>luchtbed | LID(onb,stan,agr)<br>N(soort,ev,basis,zijd,stan) | SU:NP | FE:Location |
| slaapt | WW(pv,tgw,met-t) | HD:V | (Sleep) |
| op | VZ(init) | | |
| zich | VNW(refl,pron,obl,red,3,getal) | | |
| wel | BW() | | |
| heel<br>erg<br>fijn | ADJ(vrij,basis,zonder)<br>ADJ(vrij,basis,zonder)<br>ADJ(vrij,basis,zonder) | MOD:AdvP | FE:Depictive |
| . | LET | | |
| | | CNI: Sleeper | |

FIGURE 13 A CGN sentence enriched with semantic information derived from FrameNet

(20) ’Nou en die bank zit niet zo lekker (marnix als de bank waar wij
well and that couch sits not as nice (marnix as the couch where we
nou op zitten.)’
at the moment on sit
’Well, sitting on that couch is not as nice (marnix as on the couch we are sitting on at the moment.)’

<fn00729.11>

| word | pos | syn | sem |
|---|---|---|---|
| Nou | BW() | | |
| en | VG(neven) | | |
| die<br>bank | VNW(aanw,det,stan,prenom,zonder,rest)<br>N(soort,ev,basis,zijd,stan) | SU:NP | FE:Location |
| zit | WW(pv,tgw,met-t) | HD:V | (Posture) |
| niet<br>zo<br>lekker | BW()<br>BW()<br>ADJ(vrij,basis,zonder) | MOD:AdvP | FE:Depictive |
| | | CNI: Protagonist | |

FIGURE 14 A CGN sentence enriched with semantic information derived from FrameNet

that is being evoked, resp. *Sleep*, *Posture* and *Operate$_V$ehicle*. Furthermore, adjectival and nominal phrases are labelled according to the frame element they represent. In the

(21) 'de auto rijdt makkelijk'
the car drives easy
'Driving the car is easy'

<fn008066.260>

| word | pos | syn | sem |
|---|---|---|---|
| de | LID(bep,stan,rest) | | |
| auto | N(soort,ev,basis,zijd,stan) | SU:NP | FE:Vehicle |
| rijdt | WW(pv,tgw,met-t) | HD:V | (Operate_vehicle) |
| makkelijk | ADJ(vrij,basis,zonder) | MOD:AdvP | FE:Depictive |
| | | CNI:Driver | |
| . | LET() | | |

FIGURE 15 A CGN sentence enriched with semantic information derived from FrameNet

first example sentence, the subject 'een luchtbed' gets the role *Location* assigned, while the modifier gets the label *Depictive*. It should be noticed that not only verbs are FEEs, other elements of the sentence can also be a FEE. FrameNet has a strategy to deal with this phenomenon, but we will ignore it in this paper. At first sight, there is no difference in the annotation of middles and other verbs. The difference lies in the presence of the Agent: if there is an FE, other than the Agent, in the subject position, then the Agent is automatically represented as CNI (in special cases it surfaces as a voor-PP). In our first example sentence, the Agent is a *Sleeper*.

## 7.7 Conclusion

The goal of this paper was to verify to which extent FrameNet could be employed to enrich a syntactically annotated corpus such as the CGN with semantic role information. To this end, we have taken a language specific phenomenon such as the Dutch adjunct middle construction, as a test case.

From our investigation, we can conclude that there is only a partial correspondence between the classification of the Dutch adjunct middle construction as attested in the literature (Peeters 1999) if it is compared with the FrameNet classification. This is due to the wide distribution of the adjunct middle verbs over the frames which goes beyond the division in three classes proposed by Peeters. However, we can distinguish a restricted set of frames that contain middle verbs, i.e. Intentionally_act, Motion and Posture indicating that FrameNet is suitable for making linguistic generalizations.

On the other hand, when it comes to frame elements this is not the case, since the traditional Agent role gets many different labels across various frames. Other frame elements are more constant across frames although their definitions are not always the same. As for the labelling of the adjuncts which surface in subject position, we also see a more fine grained division in FrameNet than that postulated in the literature. More generally, FrameNet reaches a level of granularity in the specification of the semantic roles which might be desirable for certain applications (i.e. Question Answering). However, it makes automatic annotation of semantic roles rather impossible and might even raise problems with respect to uniformity of role labelling even if human annotators are involved.

Furthermore, incompleteness constitutes a serious problem, i.e. several frames and relations among frames are missing mainly because FrameNet is still under development. Adopting the FrameNet lexicon for semantic annotation means contributing to its development with the addition of (language specific) and missing frames. Incompleteness is also a problem within the CGN since at its present stage the corpus lacks information about subcategorization, which, however, can be inferred on the basis of the dependency structure.

In our study, we have assumed that the FrameNet classification even though it is based on English could be applicable to Dutch as well. Although Dutch and English are quite similar, there are differences on both sides. For example, in the case of the Spanish FrameNet it turned out that frames may differ in their number of elements across languages (cf. (Subirats and Petruck 2003) and (Subirats and Sato 2004)).

On the basis of our preliminary investigation, we can conclude that FrameNet offers a way to correctly classify the Dutch adjunct middle verbs. Even though some problems have emerged, our test case indicates that the FrameNet lexicon can be employed to semantically annotate the Spoken Dutch Corpus. However, we need to verify in more details to which extent the English frames translate into Dutch frames. In this respect, we can benefit from results from projects like SALSA ((Erk et al. 2003)) where FrameNet is used to annotate the German Tiger Corpus ((Brants et al. 2002)).

In our study, we have assumed the Spoken Dutch Corpus as our basis. We still have to assess whether the FrameNet lexicon is also suitable for the semantic annotation of the written Dutch corpus which is being developed within the D-Coi project which employs the Alpino parser to add the syntactic layer of annotation to the corpus. Furthermore, we did not yet discuss the possibility of applying the PropBank approach to role assigment (Kingsbury, Palmer, and Marcus 2002). This approach is essentially corpus based and syntax driven and while the more semantic driven FrameNet approach which is based on a network of relations between frames. Another difference is that in PropBank verbs are not categorized under a specific concept but for each verb its *sense(s)* are classified under a framefile and the set of possible semantic roles is more restricted. In this respect it is worth noticing that the PropBank framefiles are quite different from the FrameNet frames. In our follow-up study (Monachesi and Trapman 2006) we examine in more detail the differences and similarities of the two approaches and the possibilities they provide for semantic annotation. We also consider in this paper the reconciliation of the two since this might result in a scheme which includes ontological information, without having a too fine grained list of possible roles.

## References

Ackema, P. and Schoorlemmer, M. (1993). The middle construction and the syntax-semantics interface *Lingua 93, pp. 59-90*.

Ackema, P. and Schoorlemmer, M. (1995). Middles and Nonmovement, *Linguistic Inquiry 26, pp. 173-197*.

Brants, S., Dipper, S., Hansen, S., Lezius W. and Smith G. (2002). The TIGER Treebank, *Proceedings of the Workshop on Treebanks and Linguistic Theories*. Sozopol.

Erk, K., Kowalski, A., Pado S. and Pinkal, M. (2003). Towards a resource for lexical semantics: A large German corpus with extensive semantic annotation. In *Proceedings of ACL 2003*. Sapporo.

Fagan, S. (1992). The syntax and semantics of middle constructions. Cambridge University Press.

Fellbaum, C. (1986). On the middle construction in English. Bloomington, Indiana: Indiana Univ. Linguistics Club.

Fillmore, C.J., Baker, C.F. and Sato, H. (2004). FrameNet as a net, *Proceedings of LREC*, Lisbon, Elra. Volume 4, pp. 1091–1094.

Haeseryn, W., Romijn, K., Geerts, G., De Rooij, J. and Van den Toorn, M.C. (1997). Algemene Nederlandse Spraakkunst. Tweede, geheel herziene druk, 1997. Groningen/Deurne, Martinus Nijhoff uitgevers/Wolters Plantyn, pp. 50–55.

Hoekstra, T. and I. Roberts (1993). Middle constructions in Dutch and English, *Knowledge and Language*. Kluwer Academic Publishers, Dordrecht, pp. 183–220.

Johnson, C.R. and Fillmore C.J. (2000). The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure, *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, Seattle WA, pp. 56–62.

Johnson, C.R., Fillmore, C.J., Petruck, M.R.L., Baker, C.F., Ellsworth, M.J., Ruppenhofer, J., and Wood, E.J. (2002). FrameNet: Theory and Practice (e-book), `http://framenet.icsi.berkeley.edu/book/book.pdf`

Kingsbury, P., Palmer, M. and Marcus, M. (2002). Adding Semantic Annotation to the Penn TreeBank, *Proceedings of the Human Language Technology Conference. HLT-2002*. San Diego, California.

Monachesi, P. and Trapman, J.R. (2006). Merging FrameNet and PropBank in a corpus of written Dutch, *Proceedings of the workshop Merging and Layering Linguistic Information, LREC-2006*. Genoa, Italy.

Oostdijk, N., Goedertier, W., Van Eynde, F., Bovens, L., Martens, J.P., Moortgat, M. and Baayen, H. (2002). Experiences from the Spoken Dutch Corpus Project, *Proceedings of LREC-2002*, pp. 340–347.

Peeters, R.J. (1999). The adjunct middle construction in Dutch, *Leuvense Bijdragen*, jaargang 88, pp. 355–401.

Petruck, M.R.L. (1996). Frame Semantics, *in* Verschueren, J., Östman, J., Blommaert, J. and Bulcaen, C. (eds.), *Handbook of Pragmatics 1996*. Philadelphia: John Benjamins.

Subirats, C. and Petruck, M.R.L. (2003). Surprise: Spanish FrameNet!, *in* Hajicova, E., Kotesovcova, A. and Mirovsky, J. (eds.), *Proceedings of CIL 17*. Prague: Matfyz-press.

Subirats, C. and Sato H. (2004). Spanish FrameNet and FrameSQL, *Proceedings of the workshop Building Lexical Resources from Semantically Annotated Corpora, LREC-2004*. Lisbon, Portugal.

Trapman, J.R. (2005). Where FrameNet meets the Dutch Spoken Corpus: in the middle. Bachelor thesis. Utrecht University.

8

# A New Hybrid Approach Enabling MT for Languages with Little Resources

PETER DIRIX, VINCENT VANDEGHINSTE AND INEKE SCHUURMAN

*Centre for Computational Linguistics, K.U.Leuven*

## Abstract

In this paper, we combine techniques from rule-based and corpus-based MT in a hybrid approach. We only use a dictionary, basic analytical resources and a monolingual target-language corpus in order to enable the construction of an MT system for lesser-resourced languages. Statistical and example-based systems usually do not involve a lot of linguistic notions. Cutting up sentences in linguistically sound subunits improves the quality of the translation. Demarcating clauses, verb groups, noun phrases, and prepositional phrases restricts the number of possible translations and hence also the search space. The sentence chunks are translated using a dictionary and a limited set of mapping rules. By bottom-up matching the different translated items and higher-level structure with the database information, one or more plausible translated sentences are constructed. A search engine ranks them using the frequencies of occurence and the matching accuracy in the target-language corpus.

*Proceedings of the Sixteenth Computational Linguistics in the Netherlands*.
Edited by: Khalil Sima'an, Maarten de Rijke, Remko Scha and Rob van Son.

## 8.1 Introduction

Since its introduction in the 1950s, machine translation (MT) has been the holy grail of computational linguistics. The first word-by-word systems were soon succeeded by rule-based systems. Despite their numerous limitations, these systems are nowadays still the most used. Their main bottleneck is the almost infinite number of rules you have to construct to get a good translation. Furthermore, the processing time was a problem for a very long period until computers got fast enough. You also need advanced resources such as syntantic (and maybe semantic) parsers.

In the 1980s new techniques, mainly borrowed from speech recognition, gave birth to statistical machine translation (SMT). Twenty years later, there are not a lot of commercial systems available yet, although Google announced to launch an SMT system in 2007. The main disadvantages of SMT are the need of a parallel text corpus and data sparsity: the parallel corpus used is in fact never large enough! Such parallel text corpora (or *bitexts*) are hardly ever available for most language pairs and terminological domains, especially for general language. The same disadvantages apply to example-based machine translation (EBMT).[29]

The METIS-II system[30] is under development at a consortium formed by the Institute for Language and Speech Processing (ILSP) in Athens, the Universitat Pompeu Fabra in Barcelona, the Institute of Applied Information Sciences (IAI) in Saarbrücken and the Centre for Computational Linguistics (CCL) of the K.U.Leuven. This system makes use of a target-language corpus only, and therefore by-passes the bitext problem. On the other hand, it needs a bilingual dictionary, a limited set of translation rules and a basic (shallow) source-language analysis.

The rationale for this approach is that for many, especially smaller, EU languages little digital resources are available (cf. the BLARK initiative[31]). Parallel corpora for language pairs of which at least one language does belong to this set of smaller languages are very scarce (even when the other language is English).

The fact that there are huge amounts of documents waiting to be translated, involving all kinds of language pairs for which only limited resources are available (e.g. no full parser, no large enough parallel corpus) made us investigate whether a machine translation technique can be developed for use under these conditions. So, although for the languages involved in the METIS-II project these more advanced tools are available[32], we refrain from using them in order to mimic the situation lots of *low-resource* languages are faced with. Hence, we are not claiming that our approach is better than the ones generally used in SMT and EBMT, when a large (huge) parallel corpus for a specific subdomain and a

[29]For a description of recent techniques, see Carl and Way (2003)

[30]Supported by the 6th European Framework Programme, FP6-IST-003768. It is the successor of the METIS-I project (Dologlou, Markantonatou, Tambouratzis, Yannoutsou, Fourla and Ioannou 2003), which confirmed the feasibility of this approach.

[31]For Dutch, a report was drawn up by Daelemans and Strik (2002).

[32]But note that even for the pair Dutch-English a large parallel corpus in the general domain does not yet exist!

specific language pair is available. The only *advanced* resource we are using is a bilingual dictionary, consisting of lemmas and their part of speech in both languages.[33]

The introduction of mapping rules could resolve some linguistic issues that arise with SMT and EBMT techniques. The combination of rule-based and statistical/example-based methods leads to a *hybrid* system, which seems the way to go (Thurmair 2005), to avoid the intrinsic obstacles of both the statistical and rule-based methods. The system uses a basic group of resources and a very limited set of rules, and uses the target-language corpus as the main resource for translation candidate selection and word order.

The use of this methodology enables us to construct an MT system for low-resource languages, on the condition that they possess a certain minimal set of linguistic tools, including a target-language corpus.

## 8.2 The METIS-II System

This general-domain MT system is being constructed for four language pairs: from Dutch, Modern Greek, German, and Spanish to English (Vandeghinste, Schuurman, Carl, Markantonatou and Badia 2006). Nevertheless, the system is designed in such a way that most parts are language-independent, whereas language-dependent modules can be plugged in when needed. Not all language pairs use the same resources[34], and this shows that the system can be used with a variety of resources, depending on the availability for the languages at hand, although this will have an effect on the translation accuracy. Of course, every partner institution is using its own tools for dealing with the source-language input.

At this stage, all partners developed their own expanders and search engines, albeit using the same ideas and paradigms. In addition, all are using the tagged and lemmatised versions of the target-language corpus, in this case the British National Corpus (BNC).

We will start with a general overview of the three stages in the translation process, called the language models. We will also give a short introduction to the general scoring mechanism. Next, each of the language models will be discussed in detail, describing the different modules that form the system and the way they score the building blocks of the translated sentence.

The different modules are integrated in an NLP engine that follows the flow presented in figure 16.

### 8.2.1 Three language models

The translation process is divided in three stages (see figure 16).

[33]If necessary, such a dictionary can be obtained by extending a basic vocabulary using a comparable corpus (Sadat, Déjean and Gaussier 2002), which is much easier to come by than a parallel corpus. Another possibility would be to use such a comparable corpus for translation purposes (in addition to a parallel corpus, or maybe even instead of such a corpus).

[34]Especially the Spanish team is trying to develop a system only using statistical means.

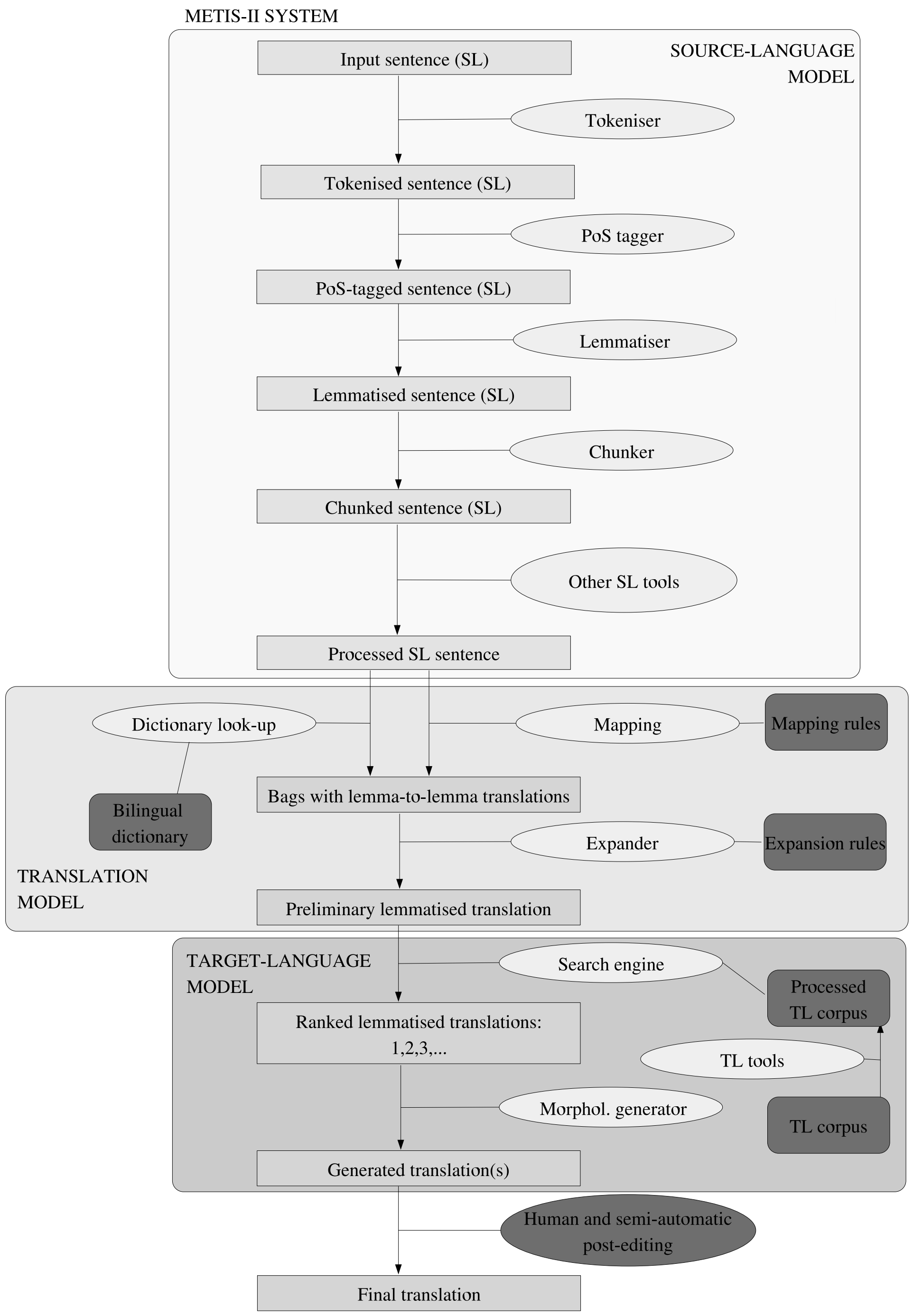


FIGURE 16 General data flow

First, a shallow *source-language model* (SLM) is constructed, using tools that analyse the input sentence. This sentence is tokenised, part-of-speech (PoS)-tagged, lemmatised and chunked into phrases. We also use additional tools like a subclause detector, and intend to use a subject detector in the near future.

Subsequently, the sentence needs to be translated. The *translation model* (TM) consists of a bilingual dictionary, a limited set of tag-mapping rules and grammatical rules to map the source structure to the target language. It enables the transition of the source-language lemmata to the target language and allows for reorganisation of the chunks in the sentence. Since modules in the SLM could generate various possible translations and structures, the system produces a list of possible translations.

The search engine compares this list with the *target-language model* (TLM), based on a target-language corpus, and chooses the (*n*) best translation(s). The fact that we are translating lemmata instead of tokens, simplifies the search by reducing the sparsity, but forces us to use a morphological generator.

Finally, this preliminary translation should be offered to a human translator for post-editing. This way preferred translations can be stored as well. A future version of the system should allow to use these preferences when scoring alternative translations.

### 8.2.2 Scoring mechanism

At all steps in the processing chain, every node in every parse tree receives a weight. The mechanism is designed as such that the joint weight is in principle one, except in the search engine, where the weight represents the matching accuracy with the corpus. If the number of possibilities is higher than a parametrisable number $N_{max}$, the *beam* is cut off before the first element with a lower weight than $N_{max}$. In cases of ambiguity where the module is not assigning any weight itself, the weight is divided proportionally over the different alternatives.

### 8.2.3 The source-language model (SLM)

For each of the relevant languages, the SLM is constructed using language-specific tools. The only condition is that the output format is compatible with the search engine. In this paper, we describe the tools used for MT from Dutch to English.

#### Basic analysing tools

The *tokeniser* is a module that identifies the separate words and punctuation marks. Every punctuation is considered a separate token. The input for this module is a source-language sentence. The PoS tagger requires the output format to be separate tokens on a different line.

The *tagger* assigns part of speech categories to the Dutch tokens, using the *CGN*-

*tagset*[35]. This tagset is based on the morphosyntactic forms of Dutch. We use the TnT tagger (Brants 2000) trained on the CGN[36], with the option that not only the best (most probable) tag, but also the alternative tags with a lower probability, are used. These are combined into several source-language analysis alternatives, with as their weights the products of the tag probabilities of the elements. These alternatives go through the rest of the translation process, as they can result in different lemmatisation and chunking, and, of course, in different translations.

The Dutch *lemmatiser* is based on the PoS tags assigned by the tagger. It uses the CGN lexicon with more than 300 000 forms (Piepenbrock 2002) to find the correct lemma for a token. For certain tokens, the lemmatisation process generates more than one token. Using the tags as extra information reduces the ambiguity substantially. The Dutch word *was* e.g. could be a noun meaning 'wax' or 'laundry'[37], but also the simple past singular tense of the verb meaning 'to be' or the simple present singular of the verb meaning 'to wash'. The PoS tag allows the lemmatiser to disambiguate the lemma.

Next, the tokenised, tagged and lemmatised sentence is *chunked* by ShaRPa 2.0[38]. ShaRPa is a rule-based shallow parser which uses a set of context-free non-recursive grammars to identify chunks. The NPs, PPs, and verb groups are identified. The heads of the phrases are marked. ShaRPa returns only one result per input, as it is a purely rule-based tool. The weight of the returned result is therefore the same as the weight of the input.

## Other analysing tools

We implemented some tools to identify the subjects and different kinds of subclauses[39] in a sentence. Even if the detection process is not perfect, it can resolve some word order problems, since the number of possible permutations is limited.

Dutch has an SVO order in main clauses and an SOV order in subclauses. This means that the subject of a sentence (and if applicable, of the subclause) is usually the first NP in the sentence or clause, unless the first NP is a temporal or spatial constituent. In this case, the algorithm chooses the next normal NP as subject. The borders of the subclauses are identified using subordinating conjunctions and the position of the verb (which is in Dutch usually at the end of the subclause). Relative clauses are identified using the relative pronouns which introduce them and the verb at the end of the clause. Initially, we also identified *om te* + infinitive constructions in Dutch, but since in the case of English, it is very difficult to delimit the corresponding infinitival phrases (because the Dutch trigger word *om* is not translated), we are not using this for the time being. Since these modules are rule-based and only give one result per input, the weights do not change.

[35]Tag set developed for the Spoken Dutch Corpus (CGN) (Van Eynde 2004).

[36]When the data from the D-CoI project becomes available, we will use the D-CoI tag set and train the tagger on the D-CoI corpus (Van den Bosch, Schuurman and Vandeghinste 2006).

[37]Two homonymous nouns with a different gender.

[38]An evaluation for Dutch can be found in Vandeghinste and Tjong Kim Sang (2004) and Vandeghinste (2005).

[39]An evaluation for Dutch can be found in Vandeghinste and Pan (2004).

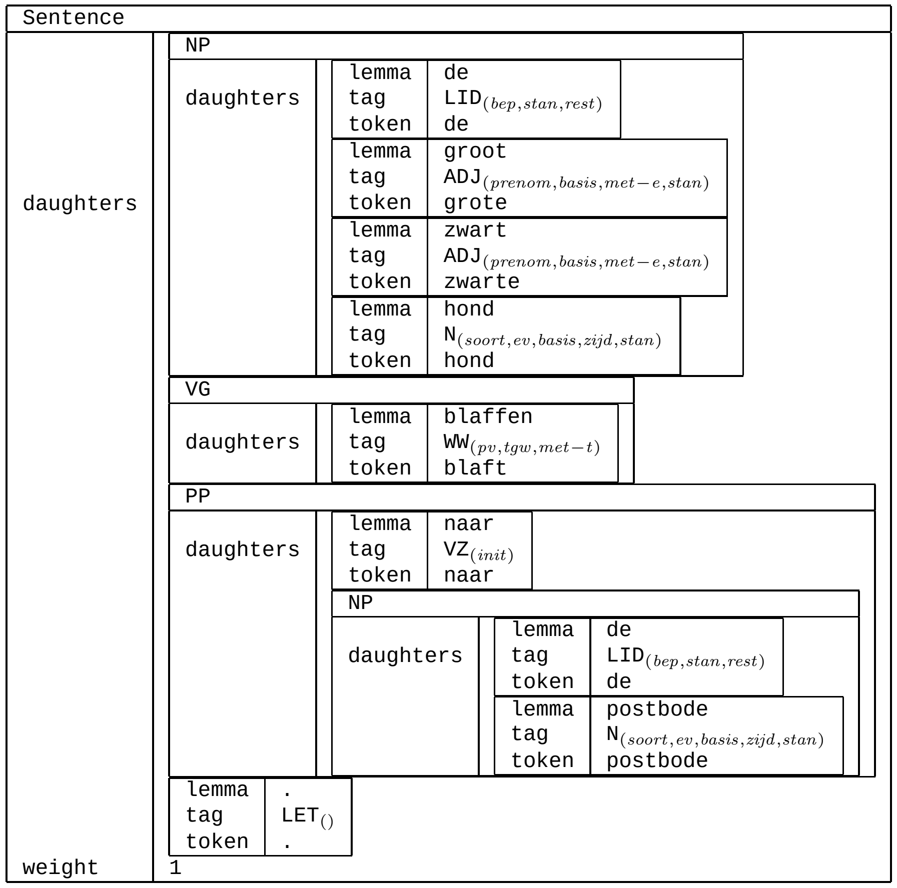
**Example**

`de grote zwarte hond blaft naar de postbode.`

$\Downarrow$

$\boxed{\texttt{SOURCE-LANGUAGE ANALYSIS}}$

$\Downarrow$


Sentence
daughters
NP
daughters
lemma de tag LID(bep,stan,rest) token de
lemma groot tag ADJ(prenom,basis,met−e,stan) token grote
lemma zwart tag ADJ(prenom,basis,met−e,stan) token zwarte
lemma hond tag N(soort,ev,basis,zijd,stan) token hond
VG
daughters
lemma blaffen tag WW(pv,tgw,met−t) token blaft
PP
daughters
lemma naar tag VZ(init) token naar
NP
daughters
lemma de tag LID(bep,stan,rest) token de
lemma postbode tag N(soort,ev,basis,zijd,stan) token postbode
lemma . tag LET() token .
weight 1

### 8.2.4 The translation model (TM)

#### Dictionary search and tag mapping

The Dutch-English dictionary was constructed using the free Internet dictionary Ergane and the Dutch EuroWordNet (Dirix 2002a). At this moment, there are about 110 000 lemma-to-lemma translations and a few hundred fixed expressions. The dictionary also contains a set of separable verbs, verbs with fixed prepositions and multiword expressions, as is shown in table 6. In these special cases, the right-hand side also contains the appropriate chunking of the English expressions. The dictionary format leaves the possibility to generalise categories and introduce extra words between the lemmas of the expression. We are currently correcting and extending the dictionary by hand.

TABLE 6 Examples of different types of dictionary entries

| SL-lemma | SL-tag | TL-lemma | TL-tag |
|---|---|---|---|
| eten | WW | eat | VV?[40] |
| weggaan | WW | go#away[41] | VV?#AV0 |
| wachten∼op[42] | WW∼VZ | wait#for | VV?#PRP |
| de#morgen | $\text{LID}_{(gen)}\#\text{N}_{(gen)}$[43] | in#the#morning | PP[in!#the#morning][44] |
| graag | BW | like∼to# <VVI>[45] | VV?!∼InP[TO0 #VVI] |

The CGN tag set is based on morphosyntactic properties of the Dutch language. It has to be mapped to the CLAWS5 tag set, which is constructed more functionally (Dirix 2002b), and which is used to tag the BNC. Over 300 CGN tags have to be mapped to about 70 CLAWS5 tags. In general, there is a many-to-one relation between the Dutch and English tags, but there are some cases where one Dutch tag has to be mapped to more than one English tag.

### Example (continued)

The dictionary entries for the words in our example sentence can be found in table 7, whereas the tag mapping rules can be found in table 8. The example sentence was introduced in section 8.2.3.

### Expansion

There are often differences in word order between two languages. Various words are inserted or deleted in translation. These differences could force the MT system to introduce additional or modified translations into the generated list of possible translations. This is the role of the *expander*.

[40]The VV? tag is the tag we use for a lemma. The question mark is an underspecification of more specific features which contain tense and number.

[41]The # sign is used to indicate consecutive separate tokens.

[42]The ∼ sign is used to indicate separate tokens which are not necessarily consecutive.

[43]The use of features to restrict the translation of a lemma to certain circumstances is allowed.

[44]When the TL-lemma is a chunk of a different type than the SL, its type needs to be indicated, as well as its head (using the '!')

[45]The usage of <VVI> indicates that, together with the information in the TL-tag column, an expander rule needs to be triggered, that places the original main verb in the <VVI> slot, and that transfers the feature information from that main verb to the feature information of *like*.

[46]In this case, the translation `grown up` is considered as one token, which contains a space. What we consider as one token depends on the decisions taken in the target-language corpus, in our case the BNC.

TABLE 7 Dictionary entries for the example sentence

| SL lemma | SL tag | TL lemma | TL tag | SL lemma | SL tag | TL lemma | TL tag |
|---|---|---|---|---|---|---|---|
| de | LID | the | AT0 | blaffen | WW | bark | VV? |
| groot | ADJ | big | AJ? | naar | VZ | according_to | PRP |
| | | great | AJ? | | | at | PRP |
| | | grown_up[46] | AJ? | | | to | PRP |
| | | large | AJ? | | | toward | PRP |
| | | major | AJ? | | | towards | PRP |
| | | tall | AJ? | | | | |
| | | in#size | PRP#NN? | | | | |
| zwart | ADJ | black | AJ? | postbode | N | postman | NN? |
| | | gloomy | AJ? | | | mailman | NN? |
| hond | N | dog | NN? | | | | |

TABLE 8 Tag mapping for the tags of the tokens in the example

| SL-tag | TL-tag |
|---|---|
| LID() | AT0 |
| ADJ(prenom,basis) | AJ0 |
| N(soort,ev,stan) | NN0\|NN1 |
| WW(pv,tgw,met-t) | VBB\|VDB\|VDZ\|VHB\|VHZ\|VM0\|VVB\|VVZ\|VDB+VVI |
| VZ() | PRF\|PRP\|TO0 |

The list of possible translations can be expanded in two different ways. The first expansion is based on the target-language corpus in order to cover the word order transitions between source and target language. The fact that the normal word order in English is adjective-noun (as opposed to noun-adjective in most Romance languages) could be derived from an English text corpus. In this case, the source-language word order has no importance for the target language.

There are also a number of issues that are source-language-dependent and hence difficult to correct when only using a target-language corpus. These modifications can be modelled with a limited set of mapping rules. An example for this case is the *do*-insertion. In English, the verb *to do* has to be inserted in almost all interrogative sentences and other cases with inversion or emphasis. Such an approach is not feasible for constructions like *ik zwem graag*, where the whole sentence structure is changed. In this case we opted for adding an entry in the bilingual lexicon with a complex lemma '*graag* + verb' in the lexicon, translated as '*like to* + verb(infinitive)' (cfr. table 6).

We use these two types of expansion in order to extend the list of possible translations that will be ranked by the search engine. We consider the input of the expander as a structured bag of bags, representing the structure of the sentence after all the source-to-target-language mapping has been applied. We want to convert this structured bag into a sentence, by resolving each subbag by searching for it in the target-language corpus (depth-first). In fact, we try to find a matching phrase that consists of all the elements of the bag. Depending on how well the corpus phrases match the bag elements, a score is calculated, resulted in a ranking of permutations, which get a final score from the search engine.

In the CCL system, the expander is currently dealing with the following list of phenomena:

1. The different parts of verb clusters are put together in one bag. In Dutch, the different parts of compound tenses can be separated by direct and indirect object, preposional phrases and even whole subclauses. The past participles and their auxiliaries are put into one bag in order to retrieve the corresponding BNC bags from the target-language corpus.
2. The literal translation of *om* in the *om te* + infinitive construction is deleted, since it remains untranslated. Again, the word *om* could be separated from the remainder of the infinitival phrase by several constituents.
3. In Dutch, the usual form of the active compound tenses is formed with the appropriate tenses of the verb *hebben* and the past participle. However, some intransitive verbs (esp. verbs of motion) are using the verb *zijn* as auxiliary in these tenses. For transitive verbs, *zijn* is used to form the passive voice of the aforesaid compound tenses. Since in English the combination *to be* and past participle is used for the translation of the Dutch '*worden* + past participle', we rewrite the literal translations '*to be* + past participle' to '*to have* + past participle' and '*to have been* + past participle'. In order not to confuse these with the passive of the non-compound tenses, we only introduce *get* and *become* as translations of *worden*. After the former rule fired, we substitute these verbs, if they are followed by a past participle, for the appropriate form of *to be*.
4. The expander is assigning the correct tags in order to translate properly the combination of a verb followed by the adverb *graag* into *to like to*, followed by a verb. We do this, using the dictionary information[47] and the fact that the tense of the original Dutch verb has to be mapped on the tense of *to like*, while the translation of the original verb gets an infinitive tag. The word order is also switched to get correct English.

### 8.2.5 The target-language model (TLM)

The consortium chose the British National Corpus (BNC) as target-language corpus. The BNC is processed analogous to the source-language input sentences: it is tokenised, PoS-tagged with the CLAWS5 tag set, lemmatised and chunked. The lemmatiser used is described in Carl, Schmidt and Schütz (2005). The corpus was chunked using ShaRPa 2.0 with an English rule set. The NPs, PPs and verb groups are identified. The head of each phrase, the sentence subject, and if applicable, the subclauses are also marked.

[47] See table 6.

## The search engine

The search engine is the nucleus of the METIS-II system. The four project partners have experimented with different types of engines. The CCL chose a bottum-up approach, as described in Dirix, Vandeghinste and Schuurman (2005) and Vandeghinste, Dirix and Schuurman (2005), and which is explained in detail in this section. The ILSP group has applied the same method in a top-down approach (Markantonatou, Sofianopoulos, Spilioti, Tambouratzis, Vassiliou, Yannoutsou and Ioannou 2005). The IAI tried the *Shake & Bake* method to select BNC constituents (Carl et al. 2005). The Spanish group finally used an *n*-gram approach (Badia, Boleda, Malero and Oliver 2005).

The search engine takes a *bag* as input. This bag can represent a chunk, a clause, or a sentence. The *elements* of a bag can be considered to be the daughters of the chunk, clause, or sentence the bag represents, but the order in which these elements have to appear in the target language has to be determined by matching the bag with the corpus.

For a given bag, we look in the corpus for a chunk, clause, or sentence (dependent on the bag level) that matches as many of the bag elements as possible. A bag element is matching a corpus element when the lemma (or lemma of the head of the constituent) matches. The accuracy of matching is quantified as follows:

$$a_i = \frac{m_i}{n_i + p_i},$$

where $m_i$ is the number of matching bag elements, $n_i$ is the total number of bag elements, and $p_i$ is the number of elements in the corpus chunk which are not in the bag (i.e. the number of insertions). When $m_i < n_i - 4$, the bag is not retained as a possible solution, because the number of insertions is too big to trust the outcome.

Not every bag alternative matches with the same accuracy, so some alternatives are preferred over other alternatives, leading to translation candidate selection when a certain combination of words occurs in the corpus.

Apart from this matching accuracy, we also take into account the relative frequency of the corpus chunk with respect to the total frequency of all corpus chunks in which the same number of elements match, as in this formula:

$$g_j = a_j \cdot \sqrt{\frac{f_j}{\sum_{k=1}{}^q f_k}},$$

where $\frac{f_j}{\sum_{k=1}{}^q f_k}$ is the relative frequency of the corpus chunk with respect to the total frequency of all corpus chunks in which $n_i$ elements match, with $k$ iterating over these elements. We take the square root of the relative frequency to make this factor less strong. The new weight for the bag $i$ matching a specific chunk $j$ is

$$w_{new,i} = w_{previous,i} \cdot g_j.$$

Once a lower level bag is solved and results in a number of translation candidates for that chunk, the head of that chunk is used at the next level, when looking for matching bag elements, and so on, until we reach the sentence level.

The corpus is indexed on the heads of chunks, so when we want to translate a chunk with a given head that is not in the corpus, we switch to *template* matching, where the same procedure is applied, but without looking at specific lemmas. Only the PoS-tags are used for matching in this case. This enables us to determine the correct word order, but is insufficient for solving the problem of different translation candidates.

## Example

TABLE 9 An example of bag matching at the NP level

| Bag Elements | | | | $n_i$ | $f_i$ | $a_i$ | $w_{new}$ | result |
|---|---|---|---|---|---|---|---|---|
| the | large | black | dog | 4 | 1 | 1.00 | 0.71 | the large black dog |
| the | big | black | dog | 4 | 1 | 0.67 | 0.47 | the big black dog |
| the | big | gloomy | dog | 3 | 5 | 0.75 | 0.37 | the big gloomy dog |
| the | great | black | dog | 3 | 2 | 0.75 | 0.23 | the great black dog |
| | | | | | 2 | 0.43 | 0.13 | the black great dog |
| | | | | | 1 | 0.27 | 0.06 | black dog the great |
| the | great | gloomy | dog | 3 | 1 | 0.75 | 0.16 | the great gloomy dog |
| | | | | | 1 | 0.43 | 0.09 | the gloomy great dog |
| the | large | gloomy | dog | 3 | 1 | 0.75 | 0.16 | the large gloomy dog |
| | | | | | 1 | 0.43 | 0.09 | the large dog gloomy |
| ... | | | | | | | | |

As shown in table 9, the bag with the four elements *the, large, black, dog* matches perfectly with a chunk from the corpus: all four elements from the bag match with the corpus ($n_i$) and all elements from the corpus chunk are matched with bag elements. This results in $a_i = 1$. There is another bag for which four elements match with the corpus, but here, the corpus chunk contains more information than the bag elements, resulting in an $a_i = 0.67$. Both these chunks occur once in the corpus, so we multiply their matching accuracy with the square root of the relative frequency with respect to all bags that match with the same $n_i$ ($f_{rel,i} = \sqrt{\frac{1}{2}} = 0.71$), resulting in the values in column $w_{new}$.

## The morphological generator

Up to now, the translated sentence consists of lemmata. This means that the correct morphological forms still have to be generated. The algorithm of the English lemmatiser used for the BNC is reversible and hence, could be used as a morphological generator (Carl et al. 2005). The tag coming from the tag-mapping rules allows us to resolve the specific features (like number, degree of comparison) of the tokens to be generated. The morphological generator also deals with capitalisation. The generation information is provided by a simple rule-based module that keeps a capital when it is in the target-language side of the dictionary (or equivalently, when the token has an `NP0` tag) and furthermore introduces a capital when a token is at the beginning of a sentence.

## 8.3 Evaluation

A lot of discussion is currently going on in the MT community about evaluation. Automated scores have been presented, each with their pros and cons, and with different purposes. Amongst the most famous are BLEU (Papineni, Roukos, Ward and Zhu 2001), NIST (Doddington 2002), WNM (Babych 2004), Test Point Method (Yu 1993), X and D-score (Rajman and Hartley 2001), and the Entropy Method (Liu, Hou, Lin, Qian, Zhang and Isahara 2005). We will present BLEU scores, as they have become a kind of standard in MT, and are easy to calculate, but they only correlate moderately (this holds for all automated scores) with human judgements about fluency and adequacy, and should be taken with a grain of salt.

Two evaluations have currently been performed: an evaluation on 150 sentences in which the source language independent parts were tested, and a second evaluation in which 50 sentences went through the whole processing chain from Dutch to English.

### 8.3.1 Source-language-independent evaluation

The search engine was tested on sentences coming from Dutch, Greek, and Spanish, on which source-language analysis was performed and manually corrected. This resulted in 150 bags of bags which we used as input for the search engine. The average BLEU score was 0.2117.

A detailed error analysis led to the introduction of the expander. The expander was taken into account in the full chain evaluation of the next section.

### 8.3.2 Full chain evaluation

We also tested our system on the full chain of processes which has to be performed in our translation system. This resulted in a BLEU score of 0.2354.

Note that not all phenomena which occur in the test set have been implemented, and that there is still a lot of room for improvement. The sentences in the test set were not selected randomly, but they are selected from newspaper material and are made sure to cover a number of different known difficulties in automated translation.

A detailed error analysis showed that our source-language analysis returned the correct result as best result in 54% of the cases. In an additional 16% of the cases the correct result was the second best. Tagging was correct for 76% of the test sentences. Tagging errors almost always lead to chunking errors. A weak point in the chunker is the scope of coordination, which is very hard to determine using context-free techniques, and which often leads to inaccurate chunking. In some cases the system finds the most plausible translation using the second-best tag path, instead of the best tag path.

Nevertheless, there is room for improvement both in source language analysis and in the translation engine. In the near future, we intend to switch to the D-CoI tagger for Dutch (Van den Bosch et al. 2006), improve our chunking grammars, and add some more rules to our expander so that more MT phenomena can be solved.

## 8.4 Conclusion and future

As said before, the actual goal of the METIS-II project is not to construct a better MT system than the currently existing ones, but to find a methodology to simplify the construction of new MT systems and language pairs, especially for lesser-used languages and domains where no parallel corpora are available. After the success of METIS-I, we have started to improve the quality of the translations. The first step was introducing chunking in order to increase the probability of finding an exact match in the target-language corpus.

Basically, translation is done by the bilingual dictionary and the tag-mapping rules. However, in order to provide the search engine with better translation candidates to rank, an expander was introduced. The expander uses a very limited rule set in order to rewrite or expand the candidates provided by the dictionary and the tag mapping.

The results generated by the system up to now, can be seen as a baseline for future improvements of the system. The BLEU score of 0.2354 can be augmented in a lot of ways and currently, we are working on correcting generic errors that happen to occur in our test set.

The Dutch-English dictionary is being revised at this time. The chunking rules of ShaRPa 2.0 can be refined, both for Dutch and English. The subject position is still not used in the target-language model but is in the process of being integrated. Postprocessing modules can be constructed to correct generic errors introduced by the search engine.

Finally, we need to develop some post-editing modules. The proposed translation(s) will be presented to a human editor, who can choose the best option and correct mistakes still there. We can use these corrections as an extension to the target-language model.

A more elaborate test set needs to be created, so more extensive evaluations can be done, using automated metrics like BLEU, NIST, and Levenshtein, and human judgment scores.

## References

Babych, B.(2004), Weighted N-gram model for Evaluating Machine Translation Output, *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics*, Birmingham.

Badia, T., Boleda, G., Malero, M. and Oliver, A.(2005), An *n*-gram approach to exploiting a monolingual corpus for Machine Translation, *Proceedings of MT Summit X, Workshop on EBMT*, Phuket, pp. 1–7.

Brants, T.(2000), TnT – A Statistical Part-of-Speech Tagger, *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Carl, M. and Way, A. (eds)(2003), *Recent Advances in Example-Based Machine Translation*, Kluwer Academic Publishers, Dordrecht.

Carl, M., Schmidt, P. and Schütz, J.(2005), Reversible Template-based Shake & Bake Generation, *Proceedings of MT Summit X, Workshop on EBMT*, Phuket, pp. 17–25.

Daelemans, W. and Strik, H.(2002), Het Nederlands in taal- en spraaktechnologie: prioriteiten voor basisvoorzieningen, Report by order of the Dutch Language Union.

Dirix, P.(2002a), The METIS Project: Lexical Resources, Internship Report, K.U.Leuven.

Dirix, P.(2002b), The METIS Project: Tag-mapping Rules, Paper, K.U.Leuven.

Dirix, P., Vandeghinste, V. and Schuurman, I.(2005), METIS-II: Example-based translation using monolingual corpora – System description, *Proceedings of MT Summit X, Workshop on EBMT*, Phuket, pp. 43–50.

Doddington, G.(2002), Automatic Evaluation of Machine Translation Quality using N-gram Co-occurrence Statistics, *Proceedings of the 2th Human Language Technologies Conference*, San Diego, pp. 128–132.

Dologlou, Y., Markantonatou, S., Tambouratzis, G., Yannoutsou, O., Fourla, A. and Ioannou, N.(2003), Using Monolingual Corpora for Statistical Machine Translation: The METIS System, *Proceedings of EAMT-CLAW 2003: Controlled Language Translation*, Dublin City University, Dublin, pp. 61–68.

Liu, Q., Hou, H., Lin, S., Qian, Y., Zhang, Y. and Isahara, H.(2005), Introduction to China’s HTRDP Machine Translation Evaluation, *Proceedings of Machine Translation Summit X*, Phuket.

Markantonatou, S., Sofianopoulos, S., Spilioti, V., Tambouratzis, Y., Vassiliou, M., Yannoutsou, O. and Ioannou, N.(2005), Monolingual Corpus-based MT using Chunks, *Proceedings of MT Summit X, Workshop on EBMT*, Phuket, pp. 91–98.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.(2001), BLEU: a method for automatic evaluation of Machine Translation, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL, Philadelphia.

Piepenbrock, R.(2002), CGN Lexicon v. 9.3, Spoken Dutch Corpus, TST-centrale, Leiden/Antwerp.

Rajman, M. and Hartley, A.(2001), Automatically predicting MT systems rankings compatible with Fluency, Adequacy or Informativeness scores, *Proceedings of MT Summit VIII: 4th ISLE Workshop on MT Evaluation*, Santiago de Compostella.

Sadat, F., Déjean, H. and Gaussier, E.(2002), A Combination of Models for Bilingual Lexicon Extraction from Comparable Corpora, *Proceedings of the Séminaire Papillon 2002*, Tokyo.

Thurmair, G.(2005), Improving Machine Translation Quality, *Proceedings of MT Summit X*, Phuket.

Van den Bosch, A., Schuurman, I. and Vandeghinste, V.(2006), Transferring PoS-tagging and lemmatization tools from spoken to written Dutch corpus development, *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, European Language Resources Evaluation, Paris.

Van Eynde, F.(2004), Pos-tagging en lemmatisering, TST-centrale, Leiden/Antwerp.

Vandeghinste, V.(2005), Manual for ShaRPa 2.0, Internal document, K.U.Leuven.

Vandeghinste, V. and Pan, Y.(2004), Sentence Compression for Automated Subtitling. A Hybrid Approach., *Proceedings of ACL-workshop on Text Summarization*, Barcelona.

Vandeghinste, V. and Tjong Kim Sang, E.(2004), Using a Parallel Transcript/Subtitle Corpus for Sentence Compression, *Proceedings of the 4th International Confererence on Language Resources and Evaluation (LREC)*, European Language Resources Evaluation, Paris.

Vandeghinste, V., Dirix, P. and Schuurman, I.(2005), Example-based Translation without Parallel Corpora: First experiments on a prototype, *Proceedings of MT Summit X, Workshop on EBMT*, Phuket, pp. 135–142.

Vandeghinste, V., Schuurman, I., Carl, M., Markantonatou, S. and Badia, T.(2006), METIS-II: Machine Translation for Low Resource Languages, *Proceedings of the 5th International Confererence on Language Resources and Evaluation (LREC)*, European Language Resources Evaluation, Paris.

Yu, S.(1993), Automatic Evaluation of Output Quality for Machine Translation Systems, *Machine Translation* **8**, 117–126.

# List of Contributors

**Lou Boves**
Radboud University of Nijmegen
the Netherlands
P.O. Box 9103 6500 HD Nijmegen
The Netherlands
L.Boves@let.ru.nl

**Peter-Arno Coppen**
Radboud University of Nijmegen
the Netherlands
P.O. Box 9103 6500 HD Nijmegen
The Netherlands
coppen@let.ru.nl

**Tim Van de Cruys**
CLCG, University of Groningen
Groningen
The Netherlands
t.van.de.cruys@rug.nl

**Peter Dirix**
Centre for Computational Linguistics
KU Leuven
Maria Theresiastraat 21
B-3000 Leuven
Belgium
peter.dirix@ccl.kuleuven.be

**Markus Egg**
Rijksuniversiteit Groningen
Groningen
The Netherlands
egg@let.rug.nl

**Paola Monachesi**
Uil-OTS
Utrecht University
Trans 10
3512 JK Utrecht
The Netherlands
paola.monachesi@let.uu.nl

**Mark-Jan Nederhof**
Faculty of Mathematics and Natural Sciences and Faculty of Arts
University of Groningen
The Netherlands
markjan@let.rug.nl

**Gertjan van Noord**
Faculty of Mathematics and Natural Sciences and Faculty of Arts
University of Groningen
The Netherlands
vannoord@let.rug.nl

**Nelleke Oostdijk**
Radboud University of Nijmegen
the Netherlands
P.O. Box 9103 6500 HD Nijmegen
The Netherlands
N.Oostdijk@let.ru.nl

**Ineke Schuurman**
Centre for Computational Linguistics
KU Leuven
Maria Theresiastraat 21
B-3000 Leuven
Belgium
e-mail: ineke.schuurman@ccl.kuleuven.be

**Jantine Trapman**
Uil-OTS
Utrecht University
Trans 10
3512 JK Utrecht
The Netherlands
J.R.Trapman@students.uu.nl

**Vincent Vandeghinste**
Centre for Computational Linguistics
KU Leuven
Maria Theresiastraat 21
B-3000 Leuven
Belgium
vincent.vandeghinste@ccl.kuleuven.be

**Suzan Verberne**
Radboud University of Nijmegen
the Netherlands
P.O. Box 9103 6500 HD Nijmegen
The Netherlands
s.verberne@let.ru.nl

**Eline Westerhout**
Uil-OTS
Utrecht University
Trans 10
3512 JK Utrecht
The Netherlands
E.N.Westerhout@students.uu.nl

**Martijn Wieling**
Faculty of Mathematics and Natural Sciences and Faculty of Arts
University of Groningen
The Netherlands
m.b.wieling@student.rug.nl