Computational Linguistics in the Netherlands 2007

Selected papers from the eighteenth CLIN meeting

Published by
LOT
Janskerkhof 13
3512 BL Utrecht
The Netherlands

phone: +31 30 253 6006
fax: +31 30 253 6406
e-mail: lot@let.uu.nl
http://www.lotschool.nl

Cover illustration: photography: Hans van Reenen

ISBN: 978-90-78328-76-6
NUR 616

Computational Linguistics in the Netherlands 2007

Selected papers from
the eighteenth CLIN meeting
(CLIN 2007)

Edited by:

Suzan Verberne
Hans van Halteren
Peter-Arno Coppen

Language and Speech group
Radboud University Nijmegen

LOT
Utrecht
2008

Contents

Preface

In a tradition which has now already spanned about half a millennium, the Dutch-speaking community (these days split into The Netherlands and Flanders) has been strong in language skills. It might well be that the urge for linguistic activity is forced upon us by the combination of wanting to trade or otherwise cooperate on a global scale and having a pretty small language community ourselves. Or we just like language's unique mix of order and chaos. Whatever the reason, we keep attempting to be at the forefront wherever something with language is happening.

In the current electronic age, one of these places is the field of language and speech technology. Here too, we are a rather visible community. At international conferences, say ACL and Coling, there are usually relatively many Dutch and Flemish participants. Not just passive participants, but also active presenters. This would not be possible if we didn't have an infrastructure, a tradition, to support us. And part of this tradition is the annual CLIN (Computational Linguistics in the Netherlands) conference. Every year most of our computational linguistics researchers come together somewhere in The Netherlands or Flanders, to talk and listen to each other.

The conference has a very low threshold: most of the suggested presentations are given space in the programme. This allows for work in progress to be presented and — equally importantly — allows new researchers to gain experience in presenting. There are no proceedings in the sense of a volume containing all papers. Instead there is traditionally a volume of selected papers. A process of self-selection and peer review selection sieves out the best research and presents it in written form to the computational linguistics community as a whole.

This book is the eighteenth example of such a CLIN volume. It contains eight papers related to the conference held December 7, 2007 at the Radboud University in Nijmegen. The papers were selected from 15 submissions. We are very grateful to the group of reviewers who helped us in the selection process: Antal van de Bosch, Carole Tiberius, Frank van Eynde, Gisela Redeker, Gosse Bouma,

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.

Harald Baayen, Harry Bunt, Inge de Mönnik, Jakub Zavrel, Leonoor van der Beek, Maarten de Rijke, Tim van de Cruys, Walter Daelemans, Yves Peirsman, Simone Paolo Ponzetto, Toine Bogers and Wauter Bosma. As for the book as such, we also want to thank the *Landelijke Onderzoeksschool Taalwetenschap* (LOT) for their support in publishing it in their Occasional Series.

At the CLIN meeting, here were 108 participants (among who 43 graduate and undergraduate students) from 33 institutions, who enjoyed the 50 oral and 6 poster presentations. We would like to thank everybody who helped make this day a success. We cannot name them all here, but we do want to extend a special thanks to Daphne Theijssen for her help with the proceedings.

A final thanks goes to the CLIN community and especially to the core that keeps the tradition of the annual conference going: those who went before us as conference organizers and those who will follow us, hopefully to far in the future.

Nijmegen, October 2008

Suzan Verberne
Hans van Halteren
Peter-Arno Coppen

Human touch in online interaction

Q-go levert interactieve, online applicaties gebaseerd op natural language search technologie. In tegenstelling tot traditionele keyword search, kunnen mensen met behulp van Q-go's zoektechnologie vragen stellen in eigen woorden.

Op basis van unieke grammaticale, semantische analyses en opgebouwde branchespecifieke woordenboeken, begrijpt Q-go alle variaties van binnenkomende vragen. Hierdoor krijgen organisaties inzicht in wat bezoekers nu echt op hun website vragen. Dit resulteert in betere online dienstverlening en hogere conversie. Onnodige kosten op het gebied van klantcommunicatie worden snel teruggebracht.

Alle belangrijke Westerse talen worden ondersteund, inclusief talen als Catalaans. De software is volledig SaaS gebaseerd, waardoor aanvullende IT aanpassingen niet nodig zijn.

Met deze oplossingen ondersteunt Q-go banken, verzekeraars, telecombedrijven, overheidsorganisaties en logistieke dienstverleners uit binnen- en buitenland. Klanten zijn onder andere ING, UWV, Telefonica, Allianz en KLM.

Q-go werkt wereldwijd samen met onder meer Google en Microsoft. De van oorsprong Nederlandse organisatie heeft vestigingen in Amsterdam, Bonn, Frankfurt, Zürich, Madrid, Barcelona en New York.

Q-go Americas
New York
United States of America
T +212 984 0692
E SalesUSA@q-go.com

Q-go Europe
Amsterdam
The Netherlands
T +31 20 531 3800
E contact@q-go.com

www.q-go.com

Natural Language Search

1

Exploiting logical forms

Crit Cremers and Hilke Reckman
Leiden University Centre for Linguistics (LUCL)

Abstract

This paper presents a semantic setup for Dutch on the basis of deep processing. The parser and generator Delilah computes a system of logical forms that is both semantically adequate, and instrumental in processing tasks like disambiguation and inference. The logical forms are derivationally related but differ as to the level of specification and exploitability. The semantic setup is new, and is likely to be the first computed, fully specified semantics for Dutch. One of the logical forms introduces a new way of compiling out semantic dependencies. The resulting system is discussed at the crossroad of logical semantics and computational linguistics.

1.1 Logical form and grammar

Logical form we take to be that level of linguistic analysis at which lexical concepts, inferential semantics and information structure interact. The required analysis is formal, in the sense that it should account for the intersubjective and — thus — systematic aspects of sentential and lexical meaning. In particular, logical form is a level of grammatical representation at which semantic consequences can be computed — a view already expressed by Higginbotham (1985). Thus, logical form is one of the ultimate targets of linguistic analysis and the representation of what makes natural language a unique module of human cognition. Typically, logical form emanates from deep processing; there are no shallow ways to semantic

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 5–20
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: c.l.j.m.Cremers@let.leidenuniv.nl

precision.

The claim that logical form is formal by necessity, does not imply that it is bound to comply with first-order predicate logic. Predicate logic does not entertain a privileged relation to semantic interpretation: the set of meanings of a natural language is neither a subset nor a superset of the well-formed and interpretable propositions of predicate logic or their complement. It may be a helpful tool in describing certain aspects of the relations between concepts and operators in natural language; in our view, however, it is neither the target nor the anchor of natural language semantics. Logical form must be casted as a representation for which some adequate notion of semantic consequence (entailment) can be defined.

Overviewing modern grammar, it makes sense to state that the relation between logical form and syntactic structure defines the arena. That is not a trivial observation: though linguistics ranks among the elder sciences in the world, it took millennia before the proper balance between form and interpretation was questioned from a grammatical perspective. In recent times, Bertrand Russell (1949) has argued that logical form is not homomorphic to syntactic structure. Generative grammar split on the question with which aspects of meaning grammar could afford to deal (Seuren 1998, ch. 7). In the same period, Montague (1973) defined logical form by compositional interpretation, but correlated it to a pseudo-syntax. Pursuing this semantic perspective, categorial grammar (CG) started to take syntax seriously and to produce interpretations by derivation, deduction or unification (Moortgat 1997, Morrill 1994, Steedman 1996, Carpenter 1998). CG's strong generative capacity, however, challenges the boundaries of linguistic relevance. At the same time, modern grammar theories appear to converge at some formal link between structure and interpretation. The compositional nature of the main semantic configurations is by now undisputed (Heim and Kratzer 1998), that is, if we agree on the lexicon being the only source of semantic wisdom and on meaning being computable at all.

In computational linguistics, logical form is not a very common module of language processing systems. Approaches that avoid to incorporate explicit grammar, deliberately refrain from logical form. Many scholars working on such systems seem to share the scepticism about the computability of meaning that some theoretical linguists entertain. These 'agnostic' strategies govern the field, in our days. As a matter of fact, for each language that is targeted by computational efforts, the number of systems doing semantic analysis is quite restricted. Among these, computation of logical form for inference is rare. Notorious icons here are the HPSG LinGO enterprise (Copestake and Flickinger 2000), the grammars involved in the Verbmobil project (Wahlster 2000), the PARC XLE parser (Maxwell and Kaplan 1993), and DRT-related approaches, like (Bos 2001) and (Bos 2004). Non of these deals with Dutch.

Memory-based language learning as established in Daelemans and van den Bosch (2005), however, does not seem to target propositional interpretation or any other semantic level, till now. Undoubtedly, the problem for these learning approaches is twofold: there is hardly any semantic tagging from which propositional semantics can be induced — there is nothing to be stored or learned — and if it

existed, the scarceness of data might be overwhelming with regard to the subtlety that propositional interpretation for inference calls for. The text of the STEVIN program to stimulate the infrastructure for Dutch computation assigns high priority to semantic research but only refers to lexical semantic tagging (Nederlandse Taalunie 2004). Inference — the core business of computational semantics according to Bos (2004) — is not addressed.

Computing logical form is the main objective of the Dutch language processor Delilah. Delilah (http://www.delilah.eu) parses and generates sentences by applying a rigid combinatory categorial grammar, by graph unifying extensive attribute-value matrices and by computing a fully specified semantic representation. The categorial grammar can be viewed as combining Combinatory Categorial Grammar (*e.g.* Steedman (1996)) and Multimodal Categorial Grammar (Moortgat 1997), in the spirit of Baldridge and Kruijff (2003). It was defined originally by Cremers (1993) and argued to be mildly context sensitive in Cremers (1999) and as such described in Cremers (2002). It does not, however, support hypothetical reasoning but entertains syntactically rigid derivations. Thus, it avoids spurious ambiguity at the cost of giving up direct composition of fully specified semantics. As a consequence, in complex symbols syntactic and semantic types may shift. In particular, quantifiers are arguments in syntax but functors in semantics. The grammar is lexicon-driven: all grammatically relevant specifications are stored in the lexicon and subsumed to lemmas. The lexicon specifies *extended lexical units* in the sense of Poß and van der Wouden (2005). In that respect and in the way the lexicon is defined, Delilah — like most lexicalist processing systems — embodies a Construction Grammar (*e.g.* Croft (2001)). The product of parsing and generation is a comprehensive attribute-value matrix in HPSG-style, or a family of these matrices.

Semantically, a derivation in Delilah produces a value to a field in the matrix. This value is an underspecified structured storage of lambda-terms, dubbed Stored Logical Form. A separate algorithm converts these terms into a fully specified Flat Logical Form. In the remainder of this paper, the different functions of these levels of semantic representation in processing Dutch will be discussed.

1.2 Logical form in Delilah

Delilah computes three related levels of logical form: Stored Logical Form (SLF), Normal Logical form (NLF) and Flat Logical Form (FLF).

SLF is *derivational*: it is constructed by applying rules of grammar, which induces unification of complex symbols. It is *compositional* in that it expresses and reflects important aspects of the grammatical structure, and by being built derivationally. Moreover, it is *underspecified*. Important features of the interpretation are not made explicit at SLF. SLF underlies the construal of both NLF and FLF. Specialized algorithms translate SLF in NLF and FLF post-derivationally. Consequently, NLF and FLF are no longer fully compositional in the strong sense: not every aspect of these logical forms is functionally related to the grammatical structure. NLF and FLF both specify all definable features of the interpretation. They

differ in the ways of specifying semantic properties. In NLF, matters of scope and semantic dependency are encoded globally and implicitely, as in standard predicate logic. In FLF, scope and semantic dependency are compiled out and made explicit at local levels.

Underspecification tends to be seen as a felicitous feature of computed logical form. It can even be the base of inferential semantics, as was argued in Bos (2001). Bunt (2007) states that full specification has been proven to be intractable. We are particulary interested in underspecification of semantic scope. Like Bunt, Ebert (2005) argues that specifying full scope leads to a combinatorial explosion. A sentence like

A politician *can* fool *most* voters on *most* issues *most* of the time

with five (italicized) operators would lead to 5! analyses. In our view, this may be an artifact of the semantic formalism but not a property of the interpretation. For interpretation, scope is only relevant to determine which terms are (or could be) semantically dependent upon the interpretation of other terms, *i.e.* to find out where to skolemnize dynamic quantifiers. Knowledge of language tells us that there are several types of lexical and local restrictions on semantic dependency in this sense. For example, variables bound by definite quantifiers are for their valuation not dependent on any other variable. Indefinite quantifiers do not influence the valuation of each others' variables. Dynamic quantifiers do not scope out of strong islands. In Delilah, fully specified logical form is derived from underspecified SLF by an algorithm that exploits this type of grammatical knowledge. A comparable strategy is proposed in Koller and Thater (2006). Consequently, NLF and FLF are protected against spurious redundancy while being constructed, by incorporating grammatical knowledge.

Alshawi (1992) explicitly addresses the question why one should entertain multiple levels of logical form. He claims the Quasi Logical Form of the Core Language Engine to be a single level of semantic representation, with additional resolution procedures providing values for free variables. In that sense, our three logical forms also provide one single representation that is constructed in several layers. We claim that these different layers can serve distinct functions. We certainly do not claim that the logical forms arise from different semantic modes. There is one process of logical form construal unfolding at different stages.

1.3 Stored Logical Form

1.3.1 Construal

Each lexical phrase is endowed with a template — an HPSG-style attribute-value matrix — that specifies a structure *Store+Body* as the value of the lf-feature *semantics*. The *Body* is a lambda term, representing the canonical meaning of the phrase. The *Store* contains slots for the logical forms of dependent phrases: a verb stores the lf of its arguments, a preposition the lf of its complement, *etc*. The storage reflects the combinatoric patterns specified in the syntax. Thus, the store contains

lambda terms for all dependent constituents, and not just for scope-sensitive operators, as was proposed by Cooper (1983) and for early Montague grammar. For each slot in a *Store*, the variable it operates on in the *Body* is specified. Here is a general scheme for a *Store+Body* structure in a lexical template; all markers are variables, except for the predicates and relations in *Store*.

(1) *Store*: { DependentLF$_1$#X$_1$, ..., DependentLF$_n$#X$_n$ }
Body: operator$_1$^ ... operator$_q$^relation$_1$(X$_i$, X$_j$) & ... & relation$_p$(X$_k$, X$_m$)

The variables in the *Store* that stand for the logical forms of dependent constituents — *DependentLF*$_i$ in (1) — are linked inside the template to these logical forms $(1 \leq i, j, k, m \leq n)$. In general, they are instantiated by *Store+Body* structures themselves in the process of unification. As the outcome of this graph unification, the template of the computed phrase provides a *Store+Body* structure that contains all relevant logical forms of the subphrases involved but underspecifies the interaction of operators. The *relations* in (1) are, in general, constants specified in the template itself.

The store may contain elements that do not correspond to syntactic constituents. In particular, the store of a lexical verb will contain a quantifier that introduces the event characterisation of the verb. This is a typical choice that reflects the level of morphological decomposition fixed in the lexicon and the nature of the morpho-syntactic interface.

For a very simple sentence like *Every man works* the *Store+Body* structure after unifying the relevant templates for *every*, *man* and *works* looks like this:

(2) *Store:* { {*Store*: {*Store*: {∅, *Body*: λx. *man*(x)}#Q}
Body: λP.∀y. Q(y) → P(y)}#S,
∃z. event(z, *work*)#E }
Body: agentof(E,S) & attime(E,T) & time(T,present)

The conversions of the stored lambda terms into a fully specified representation is discussed in the next section about FLF. Here it is only relevant to note that representations for the quantifier, the noun, the verb and the inflectional element occur in a structured and labeled way in the sentential SLF: they are explicitly linked to constituents in the grammatical analysis. In (3), part of the lexical specification of the verb *zag* 'saw', as in *Elke man zag Henk werken* 'every man saw Henk working', is represented (slightly edited). Links between SLF and the other components of the analysis are underlined.

(3)
```
|ID:A+B
|HEAD:|CONCEPT:see
|     |PHON:ziet
|     |QLF:see
|     |SYNSEM:|ETYPE:state ....
|     |       |PERSON:or([2, 3])
|     |       |TENSEOP:at-pres
|     |       |VTYPE:semi_aux
|PHON:C
|PHONDATA:lijnop(ziet, A+B, [arg(left(1),0,D),
          arg(left(11),wh,E), arg(right(1),9,F)], C)
|SLF:{{[G&(B+H)#I,
```

```
      {{[J$ K&(B+L)#M], [], []},
      λN.∃O.quant(O, the) & property(M, O) & entails1(O, decr) & N
         & entails(O, incr)} &(B+L)#P,
      λQ.∃R.(quant(R, some) & see(R) & state(R) & entails1(R, incr)
         & Q & entails(R, incr)))&(A+B)#S], [], []},
      experiencer_of(S, I) & goal_of(S, T)& theme_of(S, P)
      & attime(S, K) & tense(S, pres)}
|SYNSEM:|CAT:s_vn
|       |CONTROL:controls(goal_of~[A+B, T], U~[B+L, T])
|       |PREDTYPE:nonerg
|       |TENSE:tensed ....
|TYPE:s_vn\0~[np^0#B+W, np^wh#B+H]/0~[vp^9#B+L]
|ARG:|ID:B+H
|    |PHON:E
|    |SLF:G
|    |SYNSEM:|CASE:nom
|    |       |CAT:np
|    |       |NUMBER:sing
|    |       |OBJ:subject_of(A+B)
|    |       |PERSON:or([2, 3])
|    |       |THETA:experiencer_of
|ARG:|ID:B+L
|    |PHON:F
|    |SLF:J
|    |SYNSEM:|CAT:vp
|    |       |EXSEM:X
|    |       |EXTTH:U~[B+L, T]
|    |       |THETA:theme_of
|ARG:|ID:B+W
|    |PHON:D
|    |SLF:X
|    |SYNSEM:|CASE:obliq
|    |       |CAT:np
|    |       |OBJ:dirobject_of(A+B)
|    |       |THETA:goal_of
```

In this example, it is clear that whatever plays an active role in SLF, is anchored to the grammatical analysis. In this sense too, SLF is compositional and derivational.

1.3.2 Application of SLF: disambiguation

As said, SLF reflects the morpho-syntactic complexity of the phrase: every constituent that contributes to the meaning of the whole phrase, is represented in the relevant *Store+Body* structure. For this reason, different SLFs of the same sentence correlate to different syntactic ways of composing a meaning for a sentence. This property of SLF can be used to determine to which extent *extended lexical units* or 'constructions' have contributed to an SLF and thus, which degree of lexical aggregation is represented by it. Normally, an interpretation with a high degree of lexical aggregation is to be preferred over an interpretation that was not or less based on extended lexical units. For the sentence *Nobody kicked the bucket till now* the reading *Till now nobody hit the bucket such that it fell* should have a considerably lower priority than the reading *Till now nobody died*. Comparing SLFs in this respect suffices in Delilah to select the most aggregated interpretation. Here is the relevant reasoning and measurement.

It is by now widely acknowledged — and it has never been denied, by all we know — that the lexicon of a natural language processing system not only specifies

single words. It must also — and maybe even predominantly — contain phrases or phrase structures that have a specialized syntax and/or semantics. At present, scholars proclaiming Construction Grammar (Croft 2001) stress the central role of non-atomic construal in natural language, up to rejecting an interesting level of syntactic combinatorics. In formal grammar and computational linguistics, however, applicants of HPSG, Categorial Grammar and Tree Adjoining Grammar always seem to have accounted for considerably more involved structures than just atomic units. Poß and van der Wouden (2005), for example, make clear that there is a huge variety of ways in which words and structures cluster to build complexes with specialized syntax or semantics. In general, lexical units that limit lexical selection or syntactic transparency will come with a meaning to which not every proper part contributes according to its proper class. Here are just some examples from Dutch:

- *hebben* with a DP can mean *possess*, among others; together with mass nouns like *honger* and *dorst* it expresses in all its morpho-syntactic varieties the property of being hungry (thirsty); the verb itself hardly contributes to this meaning; the construal *to possess hungriness (thirstiness)* cannot be banned from being parsed;

- prepositional phrases may introduce adjunctive modifiers, but very often verbs select certain prepositions to express specialized meanings, e.g. *werken op iemands DP*, meaning *affect somebody's DP*, where the DP introduces a psychological concept; parsing cannot be witheld from an adjunctive construal, but the selected meaning deserves priority;

- intransitive verbs V can be part of the so-called Dutch *way*-construction, expressing the meaning of moving to a certain position by V-ing, e.g. *to laugh oneself a way/path to DP*; the characteristic DP with the *way*-type NP does not play a role in the semantics.

In all these cases, the *Store+Body* structure representing the meaning of the extended lexical unit, is simpler than the SLF resulting from a parse that does not account for the lexicalized meaning. In particular the store will contain less distinct lambda-terms. Crucially, we do not presuppose that the semantic structure of an extended lexical unit is simpler in any logical or arithmetical sense. All we claim is that the store of an extended lexical unit will show less internal structure than its distributed counterpart. This follows from the construal of SLF. Thus, a simple measurement on the stores of SLFs may select the simplest and thus highly aggregated and most 'normal' reading.

Here is an example. Any sentence containing phrases of the type *honger hebben* will come with two analyses: one reflecting the reading 'be hungry', the other reflecting the reading 'possess hungriness'. The latter SLF will contain a store where the lambda-term for 'hungriness' is specified. The former SLF will have an empty store instead, as the lambda-term for 'to be hungry', whatever its logical complexity is, will not be stored but specified in a body field of SLF. Computing the aggregated complexity of the stores and comparing them, will identify

the former SLF as the simpler one. Since this SLF is part of a full analysis, we can select this analysis as the preferred interpretation. The case is illustrated for the sentence *Ik heb honger* 'I possess hungriness' c.q. 'I am hungry', with simplified representations of the *Store+Body* structures for the sake of transparency. The preferred reading has a simpler store than the dispreferred one.

(4) SLF 1 :{ { *ik*:*x* }, be_hungry(x) }
SLF 2 :{ { *ik*:*x*, *hungryness*:*y* }, possess(x, y) }

Delilah produces all readings permitted by the grammar. It defines for each SLF the structural complexity, *i.e.* the degree of embedding of *Store+Body* structures. For every acceptable parse, it computes a number that expresses the ratio between the structural complexity of the SLF and the total number of elements specified in the stores. This ratio marks the semantic complexity of the SLF. Parses can be ordered according to these ratios. Moreover, it computes the total number of predicates or small clauses in the bodies of the SLFs. This number is used as a secondary marker for semantic complexity, but does not necessarily discriminate between lexical units and semantically composed phrases.

Delilah is trimmed to pop up the simplest reading for a sentence by exploiting the underspecification of SLF. This method is utterly reliable, because it anchors in lexical specifications. In the analysis of the sentence *Sommige kinderen hadden weinig honger* ('some children were a little bit hungry') the *possess*-reading will be suppressed. This is because the lexicon contains an extended lexical unit relating *honger* and *hebben*, the SLF of which is simpler than the SLF constructed from independent lemmas for *to have* and *hungriness*. The SLF for the whole sentence is constructed by sheer unification (*cf.* Reckman (to appear)). Therefore, the complexity of the lexically specified SLFs is conserved in the derivation.

1.4 Flat Logical Form

1.4.1 Construal

Each SLF that passes the complexity test, is submitted to a post-derivational algorithm. This algorithm performs two tasks: it converts the SLF into a coherent formula and it compiles the additional semantic information resulting from this conversion onto each variable.

The first step involves β-reduction of implicitly typed lambda-terms, explicitly specifying scope dependencies between semantic operators like quantifiers, modalities and negation. This conversion is sensitive to scopal variation of operators, to the semantic nature of operators and to structural restrictions on scope imposed by islands or other intervention effects. In this step, a good deal of a semantic theory can be effectuated. The nature of the semantic theory in Delilah is not at stake here, however, as the formalism does not impose constraints on the theory that steers the conversion. It is noteworthy, though, that in Delilah we have chosen to represent all quantifiers by descriptions, thus integrating first and higher order forms of quantification for inference.

In the second step the information spelled out by the conversion is specified onto each occurrence of each variable by a compilation protocol that turns the semantic operators obsolete. After applying this protocol, each variable is locally sugared with information as to

- its entailment property
- the quantificational regime it is bound to
- the variables its instantiation is dependent upon.

The entailment property indicates whether the predicate of which the variable is an argument, allows for upward, downward or no entailment with respect to the variable. The specification of 'governing' variables indicates whether a variable is referentially independent or must be valuated by a choice function; the notion of variables y_i governing a variable x thus amounts to the introduction of a choice function $f(y_1, \ldots y_n)$ for x.

Here is an example. The FLF for *Every man works* is produced from an SLF like (2). The outcome of the conversion step can be compared to a standard specified logical form like (5). In (6) then, is the corresponding FLF compiled from it. Each variable is represented by a 4-tuple *variable* + *entailment index* + *quantifier index* + *governing variables*. Semantic constants are italicized, logical constants and standardized relations are in bold face.

(5) $\forall$**y** *man*(y) $\rightarrow$ $\exists$**z.** **event**(z, *work*) & **agentof**(z,y) & **attime**(z,t) & **time**(t, *present*)

(6) *man*(y+**down**+**every**+[]) &
event(z+**up**+**some**+[y], *work*) &
agentof(z+**up**+**some**+[y], y+**up**+**every**+[]) &
attime(z+**up**+**some**+[y], t+**up**+**some**+[y] , *present*)

The information encoded on the variables is compiled from an intermediate representation where lambda-terms are fully converted and scopes are specified. The index with respect to entailment—in (6): *up* and *down*—specifies the local monotony properties of the binding quantifier in the relevant domain, according to its definition as a generalised quantifier. For simple, lexical determiners this is straightforward, as these properties are lexically defined. For complex determiners like 'at least n but not more than m' they have to be computed from the composition; in many of these cases no monotony will be detected. This calculus is discussed in Reckman (to appear).

The index with respect to the binding quantifier can be complex, as the quantifier itself is complex. Yet, the intention is to classify also complex quantifiers in such a way that *e.g.* existential impact of the quantifier can be derived immediately. Note that the entailment property of the variable y varies with the domain of its quantifier: in the restriction of the universal quantifier, the variable bound by it allows for downward entailment, in the nuclear scope it gives rise to upward entailment. Referential dependency does not vary with domains.

In case an SLF at the first step in its 'spell out' gives rise to real ambiguity, this ambiguity is by definition expressed in terms of scopal dependencies, since SLF is only underspecified for that. FLF takes the form of a single conjunction, representing the complete class of readings, where each conjunct is annotated for the reading(s) it is part of. A conjunct appears twice iff a variable occurring in it may or may not be referentially dependent. To be precise, SLF (2) will yield at the first conversion step two readings, differing with respect to quantifier scope, as in (7). (8) gives the FLF: the conjunction that represents the information of all readings of the SLF.

(7) ∀**y** *man*(y) → ∃**z**. **event**(z, *work*) & **agentof**(z,y) & **attime**(z,t) & **time**(t, *present*)

∃**z**. **event**(z, *work*) & ∀**y**. *man*(*y*) → **agentof**(z,y) & **attime**(z,t) & **time**(t, *present*)

(8) *man*(y+**down**+**every**+[]):[**1**,**2**] &
event(z+**up**+**some**+[y], *work*):[**1**] &
event(z+**up**+**some**+[], *work*):[**2**] &
agentof(z+**up**+**some**+[y], y+**up**+**every**+[]):[**1**] &
agentof(z+**up**+**some**+[], y+**up**+**every**+[]):[**2**] &
attime(z+**up**+**some**+[y], t+**up**+**some**+[] , *present*):[**1**] &
attime(z+**up**+**some**+[], t+**up**+**some**+[] , *present*):[**2**]

1.4.2 Application: inference

FLF is an operator-free conjunction. All logical information is specified as indices on variable occurrences. The representation is inferential: to decide whether a certain inference is possible, it suffices to linearly inspect an FLF and for each conjunct, to decide locally whether or not it gives rise to (part of) the hypothesis.

Suppose we have some model M, with standard definitions for predicates, or an ontology Ω, with a well-defined hierarchy of predicative concepts. Assume that in M or Ω one place predicates P↑ and P↓ are defined and ordered with respect to P as P↓ ≤ P and P ≤ P↑, respectively, where ≤ expresses semantic subsumption. Then, if P(x+_+_+_) occurs as a conjunct in an FLF, inferences like the following immediately hold (f_y denotes a choice function living on y):

(9)

from φ & P(x+up+some+[]) & ψ	*infer*	∃z.P(z)	*and*	∃y.P↑(y)
from φ & P(x+up+some+[y]) & ψ	*infer*	$\exists f_y.P(f_y(x))$	*and*	$\exists f_y.P\uparrow (f_y(x))$
from φ & P(x+down+no+[]) & ψ	*infer*	¬∃z.P(z)	*and*	¬∃z.P↓ (z)
from φ & P(x+down+every+[]) & ψ	*infer*	∀z.P↓(z)	*and*	∃z.P ↓(z)
from φ & P(x+up+every+[]) & ψ	*infer*	∀z.P↑(z)	*and*	∃z.P ↑(z)
from φ & P(x+down+many+[y]) & ψ	*infer*	$\exists f_y.P(f_y(x))$	*and*	$\exists f_y.P\downarrow (f_y(x))$

That is, for our sentence *Every man works* and well-defined concepts *person*, *young man* and *do something* in M or Ω, each of the following entailments can be made without any additional calculation:

(10) someone does something

every man does something
all young men work
some man does something
some man works

In the inferences with respect to the universal quantifier, we assume with Seuren (2006) that universal quantification in natural language should not be modeled with empty restrictions. Every inference machine, however, may implement its own theory on semantic entailments and logical consequence. That is not at stake here. The point is that FLF offers semantic representation at a level of specification that obsoletes deep inspection of the formula for the sake of automated inference.

Under these conditions, generalized entailment according to the definition of the RTE challenges (Dagan et al. 2005) can easily be defined for FLF.

(11) *generalized entailment*
A text T, represented in FLF entails hypothesis H represented in FLF *iff* H is a finite conjunction of clauses h_i and for each h_i there is a cover φ & p & ψ of T such that *p* entails h_i

Clearly, for each 'small clause' h_i the *p* it needs for entailment is given by its main predicate. We assume that each predicate is explicitly ordered—ontologically or lexically—to not more than a finite number of other predicates. Under that assumption, for each h_i only a linear inspection of T's representation suffices to find a possible antecedent to an entailment relation. If that possible antecedent is found, entailment is checked by inspecting the entailment tables. No serious computing is involved.

There is one complication. Once a partial hypothesis living on a variable X is established, other instances of X in the hypothesis can only be derived under the same set of dependency constraints. This requires some bookkeeping, but does not frustrate the computation. Below, a simple example is given of a one line text *Elke man zag Henk werken* and a hypothesis *Iemand zag iets* 'somebody saw something'. Each part of the hypothesis is inferred from the co-marked clause in the text's FLF. (We assume that *man* is subsumed under *person* and *property* under *thing*, temporal information is ignored.)

(12) FLF(*Elke man zag Henk werken*) =

```
△ man(A+decr+every+[]) &
□ property(B+decr+the+[]) &
  exists(C+incr+henk+[]) &
  work(D+incr+some+[B]) &
  event(D+incr+some+[B]) &
  agent_of(D+incr+some+[B],C+incr+henk+[]) &
  attime(D+incr+some+[B],E) &
♢ see(F+incr+some+[A]) &
♡ state(F+incr+some+[A]) &
♣ experiencer_of(F+incr+some+[A], A+incr+every+[]) &
  goal_of(F+incr+some+[A], C+incr+henk+[]) &
```

```
♠ theme_of(F+incr+some+[A], B+incr+the+[]) &
  attime(F+incr+some+[A], H) &
  tense(F+incr+some+[A], past)
```

FLF(*Iemand zag iets*) =

```
△ person(Z+incr+some+[]) &
□ thing(Y+incr+some+[]) &
♣ experiencer_of(X+incr+some+[], Z+incr+some+[]) &
♢ see(X+incr+some+[]) &
♡ state(X+incr+some+[]) &
♠ theme_of(X+incr+some+[], Y+incr+some+[])
```

On inspection of this type of inference, one can easily conclude that *partial entailment* is defined straightforwardly on the base of (11): a hypothesis $<h_1,...h_n>$ is partially entailed by T if some h_i is entailed and some h_j is not. Partial entailment may induce a whole range of qualifications of the relation between a hypothesis and a text. In particular, an inference engine for FLF combined with an lf-driven generator can construct those subhypotheses that are entailed.

1.5 Normal Logical Form

In our approach, there is no special grammatical task for NLF. It is casted as enriched predicate logic, in the sense that it should be able to accommodate other than strict logical operators and quantifiers. For comparison, we give for a sentence like *Elke man zag Henk werken* SLF, FLF and NLF respectively.

(13) SLF

```
{{[ {{[{{[], [], []}, λI.man(I)}$J&(9+11)#K], [], []}, λL.∃J.quant(J,
every) & K & entails1(J, decr) & L & entails(J, incr) }&(5+9)#M,
{{[{{[ {{[], [], []}, λN.∃O. exists(O) & quant(O, henk) & N &
entails(O, incr)}&(16+99)#P, λQ.∃R.quant(R, some) & work(R) &
event(R) & entails1(R, incr) & Q & entails(R, incr)&(5+16)#S],
[], []}, λT.agent_of [S, P] & attime(S, T))}$U&(5+16)#V], [], []},
λW.∃X.quant(X, the) & property(V, X) & entails1(X, decr) & W &
entails(X, incr)}&(5+16)#Y, λZ.∃A1.quant(A1, some) & see(A1) &
state(A1) & entails1(A1, incr) & Z & entails(A1, incr)&(1+5)#B1 ],
[], [] }, experiencer_of(B1, M) & goal_of(B1, P) & theme_of(B1, &
attime(B1, U) & tense(B1, past) }
```

(14) FLF

```
man(A+decr+every+[]) &
property(B+decr+the+[]).[
exists(C+incr+henk+[]) &
work(D+incr+some+[B]) &
event(D+incr+some+[B]) &
agent_of(D+incr+some+[B],C+incr+henk+[]) &
attime(D+incr+some+[B],E)] &
```

```
see(F+incr+some+[A]) &
state(F+incr+some+[A]) &
experiencer_of(F+incr+some+[A],  A+incr+every+[]) &
goal_of(F+incr+some+[A], C+incr+henk+[]) &
theme_of(F+incr+some+[A], B+incr+the+[]) &
attime(F+incr+some+[A], H) &
tense(F+incr+some+[A], past)
```

(15) NLF

```
quant(A,every).[man(A) → quant(B,the).[property(B).
[quant(C,henk).[quant(D,some).[work(D) & event(D) &
agent_of(D,C) & attime(D,E)]]] &
quant(F,some).[see(F) & state(F) &
experiencer_of(F,A) & goal_of(F,C) & theme_of(F,B) &
attime(F,E) & tense(F,past)]]]
```

Just like FLF, NLF is spelled out post-derivationally from SLF. Contrary to FLF, it is neither a conjunction, nor operator-free. Just like FLF, all semantic dependencies are unambiguously and fully encoded. It is structured by logical operators. NLF is best seen as the logic label of the semantic process. NLF or an impoverished version of it can be used for standard first-order theorem proving.

1.6 Discussion

Delilah computes, apart from a kind of standard full representation, two related but procedural and formally very distinct levels of semantic representation: underspecified, compositional SLF and fully specified, paracompositional FLF. SLF is produced as part of the graph unification which is the kernel of the parsing and generation procedure. Therefore, its construction does not complicate the derivation. Yet, on SLF measures can be defined that express essential semantic properties of the structure and thus can be exploited for selection of readings and reduction of ambiguity. We are still in the process of developing the best and most telling measurements, but it is clear that hardboiled criteria can be found and applied in reducing ambiguity there where it lives: in the semantic structure of a sentence. Of course, not all problems of selecting readings can be handled at SLF. Pure local polysemy can not be decided upon by inspecting SLF. But the delicate balance between levels of lexical aggregation can most certainly be tracked at this level.

Because of its sensitivity to constituent structure, SLF seems a good level for generation to take off. As a matter of fact, we are working on generation algorithms that are fed with SLF and that aim at producing sentences the FLF of which entails or is entailed by the FLF derived from the input SLF. In this sense the labour division between SLF and FLF may contribute to purely meaning-driven translation, as argued in Alshawi et al. (1991) and Copestake et al. (2005). As was argued convincingly in Rosetta (Rosetta 1994), machine translation on a semantic base — this is not a pleonasm nowadays — flourishes through compositionality. FLF approaches the representation of meaning in the spirit of Minimal Recursion

Semantics (MRS) as implemented in the English Resource Grammar (Flickinger et al. 2000, Copestake et al. 2005). The main point of convergence is that MRS and FLF present full semantics while avoiding syntactic complexity of the logical form. There are some differences, however. Since FLF is derived from SLF, all constraints on the interaction of semantic operators — like weak island conditions — are integrated in this derivation. The construal of FLF thus embodies an explicit theory on the syntax-semantics interface. Furthermore, FLF encodes all scopal and inferential information directly onto the logical form, rendering inference strictly local. Finally, in FLF neo-Davidsonian event structure is adopted, for constructional reasons. Extended lexical units may be such that the modification of elements that do not contribute to the meaning of the sentence, is to be interpreted as a modification at the semantic top level. For example, in the constructions of type *honger hebben* 'to be hungry' the mass noun *honger* may carry a determiner that conveys the quantification or degree of the state: *geen honger hebben* (lit: no hunger have) means 'to be not hungry' and *weinig honger hebben* (lit: little hunger have) means 'to be a little bit hungry'. Without quantification over states and events no systematic or even finite treatment of semi-transparent extended lexical units seems possible. In this respect, neo-Davidsonian event structure leads to normalisation and homogenisation of logical form (*c.f.* Reckman (to appear)).

References

Alshawi, H., D. Carter, M. Rayner, and B. Gambäck (1991), Translation by Quasi Logical Form Transfer, *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 161–168.

Alshawi, H., editor (1992), *The Core Language Engine*, The MIT Press, Cambridge, MA, USA.

Baldridge, J. and G.-J. M. Kruijff (2003), Multi-modal Combinatory Categorial Grammar, *Proceedings of the 10th Annual Meeting of the European Association for Computational Linguistics*, pp. 211–218.

Bos, J. (2001), DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures, *in* Blackburn, P. and M. Kohlhase, editors, *ICoS-3, Inference in Computational Semantics*, Buxton, UK, pp. 117–124.

Bos, J. (2004), Computational Semantics in Discourse: Underspecification, Resolution, and Inference, *Journal of Logic, Language and Information* **13** (2), pp. 139–157, Springer.

Bunt, H. (2007), Semantic underspecification: which technique for what purpose?, *in* Bunt, H. and R. Muskens, editors, *Computing Meaning Volume 3*, Springer, pp. 55–85.

Carpenter, B. (1998), *Type-Logical Semantics*, The MIT Press, Cambridge, MA, USA.

Cooper, R. (1983), *Quantification and Syntactic Theory*, Reidel, Dordrecht, the Netherlands.

Copestake, A. and D. Flickinger (2000), An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG, *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Copestake, A., D. Flickinger, C. Pollard, and I.A. Sag (2005), Minimal Recursion Semantics: An Introduction, *Research on Language & Computation* **3** (4), pp. 281–332, Springer.

Cremers, C. (1993), *On Parsing Coordination Categorially*, PhD thesis, Leiden University, HIL dissertations.

Cremers, C. (1999), A Note on Categorial Grammar, Disharmony and Permutation, *Proceedings of the 9th conference on European chapter of the Association for Computational Linguistics*, pp. 273–274.

Cremers, Crit (2002), ('n) Betekenis Berekend, *Nederlandse Taalkunde* **7**, pp. 375–395.

Croft, W. (2001), *Radical Construction Grammar*, Oxford University Press, Oxford, UK.

Daelemans, W. and A. van den Bosch (2005), *Memory-Based Language Processing*, Studies in Natural Language Processing, Cambridge University Press.

Dagan, I., O. Glickman, and B. Magnini (2005), The PASCAL Recognising Textual Entailment Challenge, *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment* pp. 1–8, Springer, Southampton, UK.

Ebert, Chr. (2005), *Formal Investigations of Underspecified representations*, Phd thesis, University of London.

Flickinger, D., A. Copestake, and I.A. Sag (2000), HPSG Analysis of English, *in* Wahlster, W. and R. Karger, editors, *Verbmobil: Foundations of Speech-to-Speech Translation*, Springer, pp. 254–263.

Heim, I. and A. Kratzer (1998), *Semantics in Generative Grammar*, Blackwell Publishers, Oxford, UK.

Higginbotham, J. (1985), On Semantics, *Linguistic Inquiry* **16** (4), pp. 547–593.

Koller, A. and S. Thater (2006), An improved redundancy elimination algorithm for underspecified representations, *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Sydney, Australia, pp. 409–416.

Maxwell, J. T. and R. M. Kaplan (1993), The interface between phrasal and functional constraints, *Computational Linguistics* **19** (4), pp. 571–590, The MIT Press, Cambridge, MA, USA.

Montague, R. (1973), The Proper Treatment of Quantification in Ordinary English, *Approaches to Natural Language* **49**, pp. 221–242.

Moortgat, M. (1997), Categorial Type Logics, *in* van Benthem, J. and A. ter Meulen, editors, *Handbook of Logic and Language*, Elsevier, Amsterdam and The MIT Press, Cambridge, MA, USA, pp. 93–177.

Morrill, G.V. (1994), *Type Logical Grammar*, Kluwer Academic Publishers,

Boston, MA, USA.

Nederlandse Taalunie (2004), Vlaams-Nederlands meerjarenprogramma voor Nederlandstalige taal- en spraaktechnologie STEVIN Spraak- en Taaltechnologische Essentiële Voorzieningen In het Nederlands.

Poß, M. and T. van der Wouden (2005), Extended Lexical Units in Dutch, *in* van der Wouden, T., M. Poß, H. Reckman, and C. Cremers, editors, *Computational Linguistics in the Netherlands 2004*, LOT, Utrecht.

Reckman, H.G.B. (to appear), *Flat, not Shallow*, PhD thesis, Leiden University, LOT dissertations.

Rosetta, M.T. (1994), *Compositional Translation*, Kluwer, Dordrecht.

Russell, B. (1949), *The Philosophy of Logical Atomism* , University of Minnesota, Department of Philosophy, Repr. as Russell's Logical Atomism, Oxford: Fontana/Collins, 1972.

Seuren, P.A.M. (1998), *Western Linguistics. An Historical Introduction* , Blackwell Publishers.

Seuren, P.A.M. (2006), The natural logic of language and cognition, *Pragmatics: Quarterly Publication of the International Pragmatic Association* **16** (1), pp. 103–138.

Steedman, M. (1996), *Surface Structure and Interpretation*, The MIT Press, Cambridge, MA, USA.

Wahlster, W., editor (2000), *Verbmobil: Foundations of Speech-To-Speech Translation*, Springer.

2

Putting the t where it belongs

Solving a confusion problem in Dutch

Herman Stehouwer and Antal van den Bosch
Tilburg centre for Creative Computing, Tilburg University

Abstract

A common Dutch writing error is to confuse a word ending in *-d* with a neighbor word ending in *-dt*. In this paper we describe the development of a machine-learning-based disambiguator that can determine which word ending is appropriate, on the basis of its local context. We develop alternative disambiguators, varying between a single monolithic classifier and having multiple confusable experts disambiguate between confusable pairs. Disambiguation accuracy of the best developed disambiguators exceeds 99%; when we apply these disambiguators to an external test set of collected errors, our detection strategy correctly identifies up to 79% of the errors.

2.1 Introduction

Learners of languages with alphabetic writing systems and overt morphology at some point face the task of learning the language's derivational and inflectional paradigms. Mastering these paradigms is the key to producing grammatical sentences, with all the proper agreements and the proper choices of morpho-syntactic categories and their associated inflections. An important subtask to master is to choose the contextually appropriate form out of a set of paradigmatic alternatives. In the sentence *"He smiled broadly"*, the writer has to decide the proper

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 21–36
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: j.h.stehouwer@uvt.nl

forms are *He*, not *Him*; *smiled*, not another inflection of the same verb; and the adverbial form *broadly*, not the adjective *broad*. This task can be all the harder when the learner is a second-language learner whose native language uses different paradigms. A related task is faced by natural language generation systems, which need to generate the appropriate forms after having made their lexical choices.

In this paper we describe the development of a grammar checking module that could be integrated into a larger proofing system. We exemplify the approach using a notorious problem of paradigmatic choice in Dutch that takes learners some time to master. The problem, in its core a verb inflection task, still manages to confuse even experienced writers, witnessed by the many errors found on webpages, emails, and student reports. It serves as an illustration; as we will argue, the approach is easily transportable to similar morphological-paradigmatic confusability problems.

Making a forced choice in paradigmatic derivation or inflection, given a sentential context, can be straightforwardly cast as a classification task, which can be learned by a machine learning algorithm – much like selecting the appropriate sense for a polysemous word in word sense disambiguation systems. If a classifier can be trained to make the choice with zero error, then this classifier could be used as a grammar checker, applicable to any unseen text that contains a variant of the derivation or inflection the classifier is trained on. On the basis of the context, the perfect classifier could decide on which form is appropriate, given the context; if the classifier's prediction contradicts the form actually used, then the form could be flagged as a grammatical error.

Taking this route, we distinguish two views on the same problem, which lead to somewhat different machine learning experiments. From one perspective the task can be seen as a morphological generation task in which, given a context, the appropriate morphological operation needs to be triggered. For example, in deciding whether it should be *word* or *wordt* in *hij [word] thuisgebracht* (*he is (being) brought home*), a machine learner would be faced with the particular context *Hij . . . thuisgebracht*, the unresolved word *[word]*, and the task to classify this situation. Although this definition of the task seems straightforward (it does not presuppose any linguistic analysis of the data, as it operates just on surface cues found in wordforms), and the type of task appears quite central to natural language generation, there is not an abundance of literature approaching this task in this knowledge-free way. In (Yarowsky and Wicentowski 2000) a successful "minimally supervised" morphological analyzer is introduced which also works well on highly irregular forms. Similar work is described in (Schone and Jurafsky 2001), who focus on the inflection system of Dutch, English, and German. More generally, our definition is related in spirit to machine-learning approaches to morphological generation tasks at the word level, such as in past tense generation with English verbs (Mooney and Califf 1995) or diminutive inflection on Dutch nouns (Daelemans et al. 1997b), except that our predictive features come from the neighboring sentential context, not from the word itself.

Alternatively, and this is our second perspective, the task can be seen as a multitude of pairwise confusable disambigation tasks, where each pair of alternatives

is the domain of one classifier, henceforth *confusable expert*. For instance, there would be a separate confusable expert for deciding whether the word *houd* or the word *houdt* would be appropriate in a given context. This is in essence the same two-class task as the first formulation, but a machine learner performing one particular confusable disambiguation will only see training examples of the two alternate wordforms. Defined as such, this definition joins a long list of earlier approaches to context-based confusable set disambiguation or the related issue of accent restoration (Yarowsky 1994, Golding 1995, Mangu and Brill 1997, Wu et al. 1999, Even-Zohar and Roth 2000, Huang and Powers 2001, Banko and Brill 2001, Van den Bosch 2006). Most of this work has concentrated on a hand-selected set of notorious confusables due to homophony (*to*, *too*, *two*) or similar spelling (*desert*, *dessert*). Yet, some of it has also touched upon inflectional or derivational confusables such as *I* versus *me* (Golding and Roth 1999), the type of task we also target.

The novelty of the approach presented here resides in the fact that we combine the two approaches in order to achieve optimal results. The key difference between the two perspectives is the specificity of the classifiers. It can be expected that the monolithic classifier can leverage from the massive amounts of examples it can be trained on, while the confusable experts may draw some advantage out of the fact that there may be specific lexical contextual markers for individual confusables (such as particular verbs that a confusable adverbial form will typically follow) that would go unnoticed by the monolithic classifier. By combining the two approaches, a best-of-both-worlds solution may be reached.

A powerful help to both approaches, and a strong point for the reusability of the approach in general, is that large amounts of labeled examples can be gathered freely from digital text corpora, without the need for annotation. The very occurrence of any word belonging to a confusable pair is effectively an example in context, labeled with one of the two possible outcomes. The only disadvantage of this "free lunch" is that if a text corpus in fact contains a writing error (e.g. a text may contain the incorrect *hij *word thuisgebracht*), this example will either turn up as a falsely labeled training example, potentially causing the resulting classifier to err later on, or as a falsely labeled test example, causing errors in statistics of false hits and misses. We will not be able to solve this, nor measure the size of the effect of this hidden noise. However, this hidden factor does not invalidate the target of our study, which is to measure the error detection capabilities of our trained classifiers on actual, manually collected cases of confusion.

The latter is the reason why our evaluation focuses on two aspects: (1) First we measure how accurately our classifiers can decide between the two outcomes of the task – at this point, we ignore whether our test data might actually contain corpus errors; (2) Second, we also confront our best-performing classifiers with pre-compiled lists of confusable errors in context, collected on the web, to see what portion of these errors are actually recognized as such, providing a sample-based error detection recall estimate.

It is unlikely that we would be able to develop perfect classifiers, but low error rates should be attainable. The goal of this paper is to test whether classifiers can

be trained to classify at such low error rates that they can detect confusions well beyond the baseline level.

This paper is structured as follows. First, in Section 2.2 we introduce the task in some detail. Section 2.3 describes the systems developed. Section 2.4 reports on the experiments we performed; the results of the experiments are given in Section 7.4, and their wider implications are discussed in Section 2.7.

2.2 The confusion of Dutch words ending in -d versus -dt

The frequently occurring confusion of words ending in *-d* or *-dt* in Dutch is rooted in a verbal inflection choice, but extends to other forms as well. We first focus on the verbal inflection issue. A notorious problem in Dutch is the normal singular present tense *-t* inflection of verbs of which the stem ends in *-d*. The problematic issue is that the inflected form is pronounced the same as the form without the *-t*, as word-final *-d* and *-dt* are both devoiced and pronounced as /t/ in Dutch. So, the frequent verbs *word* and *wordt* (*become* and *becomes*) are both pronounced /wOrt/. When wordforms are homophonic, they tend to get confused often in writing (cf. the situation with *to*, *too*, and *two*, or *there*, *their*, and *they're* in English) (Sandra et al. 2001, Van den Bosch and Daelemans 2007).

As a rule, the singular second-person and third-person forms of Dutch present-tense verbs receive a *-t* inflection: *ik loop* (I walk), *jij loopt* (you walk), *zij loopt* (she walks). Besides exceptions with high-frequency irregular verbs such as *zijn* (to be), an important subregularity is that when the second-person subject occurs after the verb, e.g. in questions, the *-t* inflection is not realized (compare *jij loopt – you walk*, to *loop jij? – do you walk?*) (Geerts et al. 1984).

Although this verbal inflection issue lies at the root of the *-d* versus *-dt* confusion problem, it also affects words with the same form that are not present-tense verbs. Verb forms ending in *-d* often occur with that same form as a singular noun (e.g. *strand* can occur as a noun, e.g. *het strand – the beach*, or as a verb, *ik strand – I get stuck*), and in a similar vein present-tense verbs in *-d* can have the same form as the past participle form of the same verb (e.g. *ik verwoord het – I phrase it*, versus *ik heb het verwoord – I phrased it*). Due to the homophonic confusion and due to cognitive errors of language learners and writers in general, also these forms may occasionally get an incorrect *-dt* ending.

2.3 System architecture

2.3.1 Example generation

We collect examples from large unannotated tokenized texts by (1) finding all words ending in *-dt*, and (2) finding all corresponding alternatives having the same character string up to *-dt*, ending in *-d*. We did not filter out nouns and past participles, to keep the method as general and knowledge-free as possible, as argued in the previous section.

Figure 2.1: Generation of one Dutch windowed and labeled example.

For our experiments on the Dutch task we use the Twente News Corpus[1], which contains about 365 million tokens. The corpus is part-of-speech tagged with a state-of-the-art tagger for Dutch[2] at an estimated accuracy of 96.5%. Alternatively, the corpus is also tagged with unsupervised part-of-speech tags generated with Biemann's Chinese Whispers algorithm (Biemann 2007) on which an MBT tagger was trained[3]. In the experiments described in more detail in Section 2.4, we compare three experimental conditions in which we either do not use tags, or use Biemann's unsupervised tags, or the tags generated by the language-specific part-of-speech tagger. In the corpus we find 2,975,185 examples of *-d* versus *-dt* wordforms. Words ending in *-d* are in the majority (65%), which is partly due to the fact that the matching nouns and past participle forms all end in *-d*.

To turn the words in their sentential contexts into proper machine learning examples, a window with a limited focus of three words to the left, and three words to the right is imposed on the sentence, centering on the focus word. The focus word itself, containing the answer to the problem, is necessarily masked; we remove the information to be predicted by reducing the focus word to the shorter of the two forms. Figure 2.1 illustrates the conversion of an occurence of a Dutch confusable in context, to labeled examples. In the experimental conditions in which we also include part-of-speech information, the tags of all context words in the input window are included. However, the tag of the focus word is masked, as the focus tag reveals the outcome. Any feature that would unequivocally vote for the class suggested by the focus wordform would be counterproductive in a system that is to be used to detect that the focus wordform is in fact contextually inappropriate.

In order to be able to extract these local contexts we must first identify which words are part of the confusable problem. We used the following simple rule to build lists of confusable pairs: We select all words of the form *[stem-d]t* for which we also find at least one occurrence of the word *[stem-d]*.

[1] `http://wwwhome.cs.utwente.nl/~druid/ TwNC/TwNC-main.html`
[2] `http://ilk.uvt.nl/tadpole`
[3] See `http://ilk.uvt.nl/mbt/` – MBT is also the tagger used in Tadpole.

2.3.2 Combining classifiers

As our end goal is to develop a maximally accurate disambiguator for the two confusable tasks in order to detect actual writing errors, the question is which of the possible classifier architectures would be the most accurate. The most extreme outcomes may be on the one hand that the classifier trained on the generic morphological generation task is the best, possibly because it can be trained on most examples. On the other hand, it may turn out that an ensemble of all individual confusable experts offers the best performance. A downside of the latter architecture is that this would involve an architecture with thousands of classifiers. We decided to explore the space of combinations of generic classification with selections of word experts, under the assumption that there is an optimal selection of confusable experts that can be combined with a more generic classifier.

These ensemble systems act as gating systems: if the test word to be checked against the prediction of the system is handled by one of the selected confusable experts, the case is given to be classified to the appropriate expert; otherwise, it is passed on to be classified by the morphological generator, acting as a back-off classifier. Given such a gating architecture, the monolithic classifier can be trained in two ways: first, using all available training examples including the ones also handled by the selected confusable experts; and second, by using all training examples except those used by the specific confusable experts. The architecture of this system is visualized in a flowchart in Figure 2.2.

An important variable in this architecture is the criterion for selecting the confusable experts. In our experiments, the performance of the confusable experts in terms of error rate on 10-fold cross-validation experiments on the training set determines their selection. If an expert performs better on average than the generic morphological generator on the same 10-fold cross-validation experiment, it is selected.

2.4 Experimental setup

To provide training and test examples, we randomly shuffle all sentences in the corpus, and then divide them over a 90% training corpus, and a 10% test corpus. Our main experiment involves the disambiguation of all cases of *-d* vs. *-dt* words, in the 10% test corpus, by the ensemble system with automatically selected confusable experts. We compare the ensemble system against three simpler systems:

1. A baseline system that predicts the overall most likely outcome; this amounts to always predicting *-d*;
2. A confusable expert baseline system that predicts for each confusable expert the most likely outcome, which may deviate from the overall most likely outcome (analogous to the “most frequent sense” baseline in word sense disambiguation);
3. The monolithic classifiers in isolation, i.e., the ensemble system without the confusable experts.

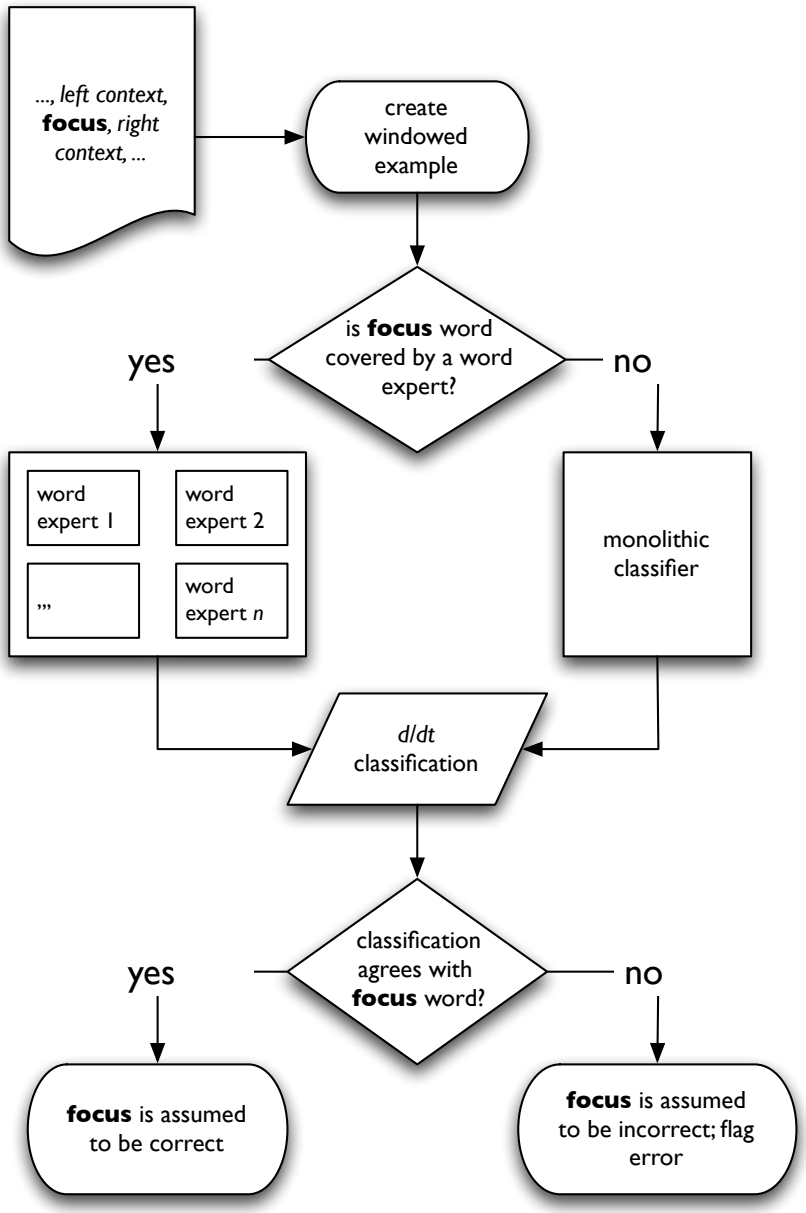

Figure 2.2: Flowchart of the grammar checking process with the ensemble system of confusable experts and the monolithic classifier. A potentially confused word in context, encoded as a windowed example, is classified by either a confusable expert or the monolithic classifier. If the classification does not agree with the original wordform in focus, an error is flagged.

While the monolithic classifier has a total of 2,646,369 cases to train on (the number of *-dt* and *-d* tokens in the training corpus), some of the confusable experts will have but a handful of examples. The most frequent confusable pair, *word* versus *wordt* (*become* vs. *becomes*) is represented by 679,808 training examples.

The ensemble system (cf. Figure 2.2) offers one aforementioned experimental option that we decided to vary systematically, which is whether the monolithic classifier retains the examples of the confusable experts or not (henceforth referred to as the *Keep* and ¬*Keep* variants). If a confusable expert is selected in cross validation, it could be argued that the monolithic classifier does not need to be trained on examples of the particular word pair covered by the selected confusable expert anymore, since the monolithic classifier will not deal with that particular word pair. On the other hand, it could be argued that the monolithic classifier should keep all examples including those already covered by the selected confusable experts, since they might turn out useful training examples also for other words.

Adding to the *Keep* versus ¬*Keep* variants, we expand the experiments with the ensemble systems by three variants of using part-of-speech information, resulting in a 2×3 matrix of experiments. In the first variant, language-specific part-of-speech tags are included; in the third, unsupervised tags generated by Biemann's Chinese Whispers (Biemann 2007) are incorporated in the feature vector. In the third variant, the windowed examples do not include part-of-speech information at all (such as depicted in Figure 2.1).

To allow for fast training and testing even with millions of examples, we used IGTree, a fast approximation of k-nearest neighbor classification (Daelemans et al. 1997a), as the core classifier in our experiments. IGTree compresses a set of labeled examples into a decision tree structure similar to the classic C4.5 algorithm (Quinlan 1993), except that throughout one level in the IGTree decision tree, the same feature is tested. Classification in IGTree is a simple procedure in which the decision tree is traversed from the root node down, and one path is followed that matches the actual values of the new example to be classified. If an end node is met, the outcome stored at the end node is generated as classification. If the last visited node is a non-ending node, but no outgoing arcs match with the next value to be tested, the most likely outcome stored at that last visited node is produced as the resulting classification.

IGTree is typically able to compress a large example set into a lean decision tree with high compression factors, in reasonably short time, comparable to other compression algorithms. More importantly, IGTree's classification time depends only on the number of features ($O(f)$). Indeed, we observe high compression rates: trained on almost 2 million examples of the task, IGTree builds a tree containing a mere 46,466 nodes, with which it can classify 17 thousand examples per second on a current standard computing server.

2.5 Results

We evaluate the trained classifiers in several ways. First, we measure how accurately our classifiers can decide between the two outcomes of the task on the

Table 2.1: Baseline and system error rates (%) of the monolithic classifier and the ensemble classifiers with the monolithic classifier retaining the examples of the selected confusable experts ("Keep") or not, under three POS tagging conditions. Boldfaced results mark the lowest error rate.

Baseline		**POS tagging**		
error	**System**	**supervised**	**unsupervised**	**no POS tags**
34.79	Monolithic	**0.83**	2.07	2.34
34.79	Keep	0.89	1.65	2.00
34.79	¬Keep	0.94	2.08	2.00

test set given a windowed local context. Second, we measure the capacity of our systems to detect errors in manually gathered list of confusions found on the web.

2.5.1 Disambiguation Error Rates

In order to ground our results, we first establish a naive baseline based on the majority outcome. The majority class baseline obtains an error rate of 34.79%, corresponding with the majority class occurring with 65.21% of all test instances. The less naive baseline (mentioned to the left of "Keep" and "¬Keep" in the same table) consists of a majority classifier for each word pair. The obtained error rate of this baseline is almost the same as that of the simpler baseline, however it is included here because, as we shall show later, it performs differently on the collected confusions.

Table 2.1 also reports the error rates achieved by the monolithic classifier in the line denoted with *Monolithic*. The monolithic classifier manages to attain an error rate of only 0.83% using supervised part-of-speech tags. This result is markedly better than the result obtained by the baseline (34.79%). Figure 2.3 shows the learning curve of this classifier. We can see that although classification accuracy improvement is slowing down with more training data, it would still continue to improve if we had more training data. The results of the monolithic classifier when using unsupervised part-of-speech tags or no part-of-speech tags are worse than the results with supervised part-of-speech tags (2.07% and 2.34% respectively). Still, they outperform the baseline by a wide margin.

Table 2.1 further displays the error rates measured for all of the 2×3 experiments, varying whether confusable expert data was kept or not (*Keep* vs. ¬*Keep*), and varying the information on part-of-speech tags across supervised, unsupervised and none. The results show that combining the monolithic classifier with confusable experts (selected because of their superior performance over the monolithic classifier on heldout data) does not seem to work in our experiments. The best result has a slightly higher error rate than the monolithic classifier (0.89% versus 0.83%), despite the fact that it uses 61 confusable experts to arrive at this result. It appears that the monolithic classifier has an accuracy that is hard to surpass, and

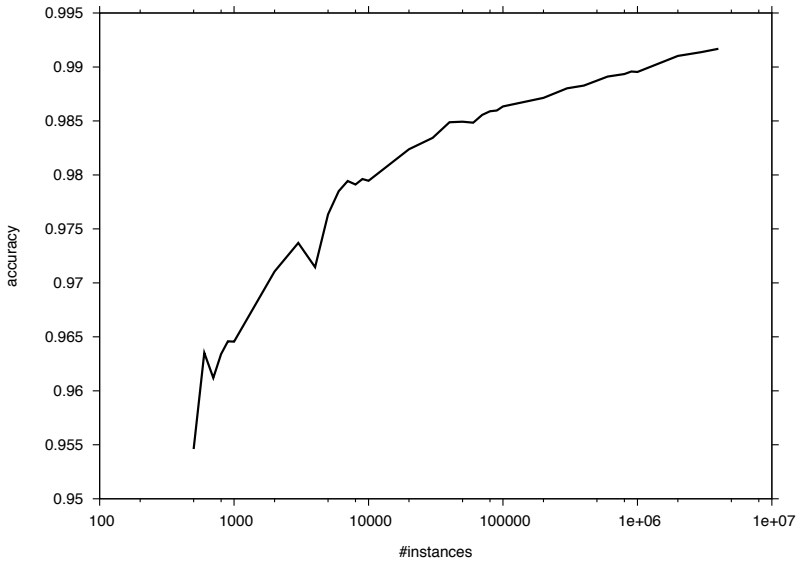

Figure 2.3: Learning curve for the monolithic classifier using the supervised POS tags as features. We plot the # of training instances used against the accuracy achieved. The x-axis is logarithmic.

may be close to a ceiling performance – note that it is trained on nearly 3 million examples. Also, estimated performances on cross-validation experiments do not provide reliable clues with respect to the performance on unseen data, apparently.

In Table 2.2 we list the top ten selected confusable experts for our best performing combination system. These top improvements concern words in the medium to low frequencies. Some high-frequency words are also selected for the combination classifier, but these show a much more modest improvement. For example, *word* versus *wordt*, the most frequent confusable, is selected, however with an improvement avarage of only 0.3%.

Focusing on the usage of supervised, unsupervised, or no part-of-speech information, we observe the following. Overall, using supervised part-of-speech tags as features yields the lowest error rates, both for the monolithic classifiers and the combination systems. The use of unsupervised tags is not favored by our results, as the error rates of these variants hardly differ from those of the systems that do not employ part-of-speech information at all.

In sum, the lowest estimated error rate is 0.83%, produced by the monolithic

Table 2.2: The top-10 better performing confusable pairs for the combination system that keeps the examples for the confusable experts also as training material for the monolithic classifier. We list the word, the number of instances it has in the training set, and the average improvement over the monolithic classifier measured through cross-validation on the training set.

Word	**# of instances**	**Improvement (%)**
onderhoud(t)	7040	.4112
spoed(t)	1083	.1994
verbreed(t)	394	.1221
ophoud(t)	950	.1198
aftreed(t)	255	.1189
strand(t)	5359	.1050
vasthoud(t)	884	.1028
aanmeld(t)	163	.0963
bijhoud(t)	238	.0955
vergoed(t)	1524	.0880

Table 2.3: Baseline and system error detection accuracy (%) on manually collected errors, by the monolithic classifiers and the *Keep* and ¬*Keep* systems, under three POS tagging conditions. Boldfaced results mark the highest detection accuracy.

Baseline		**POS tagging**		
error	**System**	**supervised**	**unsupervised**	**no POS tags**
31	Monolithic	63	63	65
36	Keep	75	68	67
36	¬Keep	**79**	63	65

classifier using supervised part-of-speech tags. This error rate appears low; yet, the question is whether these systems are accurate enough to actually spot cases of confusions to a reasonable degree.

2.5.2 Error Detection Capability

To obtain an estimate of the error detection capabilities of all 2×3 system variants, we measure their error-detection performance on confusions as they occur in texts from the web. We gathered 525 errors, bootstrapping our search using simple two-word queries such as "*ik wordt*", indicative of possible errors. We manually selected all genuine cases of errors from the search results, making sure that we sampled widely across confusable words and error patterns. All sentences with errors were tokenized and part-of-speech tagged, converted to windowed examples, and processed by the 2×3 systems, as well as by the most-frequent outcome base-

line systems. Correct error detection occurs when the system predicts a different outcome than actually present in the test example; if the system agrees with the error, the system has not detected it.

Table 2.3 lists the error detection accuracies on the collected errors. Comparing Table 2.3 to Table 2.1, one observation is that the highest error detection accuracy, 79%, is not obtained by the system with the lowest error rate. In fact, the system that uses supervised part-of-speech information and does *not* keep the training data of the confusable experts in the training data of the monolithic classifier, is the best error detector. The utility of using unsupervised part-of-speech data is not apparent from the results obtained. Again, we do not see evidence that unsupervised part-of-speech information could replace supervised part-of-speech information.

Overall, we observe that our classifiers perform markedly better than the most-frequent baselines; all systems more than double the accuracy of the baseline system (31% for the most-frequent outcome baseline, 36% for the confusable expert baseline). In terms of error reduction over the baseline, our best system is able to reduce the error by 70%.

As an additional appraisal of our approach, we compared the detection capacities of our classifiers against that of the grammar checker in the Dutch proofing tools included in the commercial word processor Microsoft Word 2003. Tested on the 525 errors, the Word grammar checker spots 88 errors, or 17%, and does not raise an alarm with the remaining errors. Our approach, with 63%–79%, clearly outperforms the Word grammar checker with respect to d/dt-error detection.

Yet, the 79% detection accuracy is considerably lower than the 99% accuracy obtained on test data, attained by the best classifiers. The discrepancy between these numbers can be explained as follows. The 99% accuracy score concerns the disambiguation of all cases of d/dt words occurring in context in newspaper text, most of which can be expected to be correct. Newspaper artices constitute professionally text that tends to be proofread and checked before publication—it has been mentioned that in texts published by the Associated Press service, 1 in 2000 words are incorrectly spelled (Church and Gale 1991). In contrast, our external test set contains only errors, occurring in text that is mostly non-professionally written, and often contains other grammar and spelling errors as well. Of the 525 errors in context, 79% can be detected correctly; it would seem that these 79% occur in contexts that resembles a context seen earlier in the newspaper text training material, while there is a 21% portion of cases where the context does not provide clues that the d/dt form in focus is actually inappropriate.

2.6 Conclusion

We presented an approach to detecting a class of confusable errors, namely those related to choices in the writing process between two similar words which differ only in their ending. We exemplified the approach on distinguishing between Dutch homophonic words ending in *-d* and their counterpart ending in *-dt*, the latter typically marking a second-person or third-person present-tense verbal inflection, such as *word* versus *wordt* (*become* versus *becomes*).

The reasoning underlying our approach is that by training classifiers on large amounts of confusable cases that can disambiguate between confusable alternatives at high accuracy, we effectively produce grammar checking subsystems that can pinpoint errors in text. By virtue of being very accurate, a discrepancy between the classifier and the actual confusable word as it occurs in running text, may well signal that the word in the text is contextually inappropriate. The question is whether we are able to train classifiers with such high accuracy (or low error rate) that they indeed can pinpoint errors with high accuracy as well. On the test problem we attained an error rate of under 1%. When applied to a manually gathered list of confusable errors found on the web, the systems are able to detect over 70% of the errors (at best 79%), markedly better than naive most-frequent outcome baselines which only correct 35% of the errors, and also better than Microsoft Word 2003 which detected only 17% of the errors.

Our systems consist of a combination of selected confusable experts combined with a back-off monolithic classifier. The selection of confusable experts is performed automatically, based on superior performance over the monolithic classifier in a cross-validation experiment on training data. In a 2×3 experimental matrix, we varied (1) whether the monolithic classifier retains or loses the training examples of the selected confusable experts, and (2) which part-of-speech information is used as input features: supervised, unsupervised, or no tags at all. On the target task, the detection of errors, we found that the best system was the ensemble system composed of selected confusable experts, where the monolithic back-off classifier did not retain the training examples of the selected confusable experts, and supervised part-of-speech tags were used in the input feature vector. We did not observe that using unsupervised tags be a proxy for having supervised tags; hence, for now we have to assume that our approach assumes the presence of a language-specific part-of-speech tagger.

To conclude the paper we discuss the novelty and generality of the approach, and we critically review the issue of evaluation. First, the core novelty of our approach lies in the fact that our systems use an ensemble architecture. In the ensemble, some classifiers are confusable-specific (the confusable experts), while one monolithic classifier generically solves the problem. Our systems are gating systems (cf. Figure 2.2) that use the monolithic classifier as a back-off classifier for solving those cases not covered by the selected confusable experts. Earlier approaches either focused on the monolithic solution (Yarowsky and Wicentowski 2000) or on individual confusable expert submodules (Golding and Roth 1999). Our results on error detection show that the monolithic approach can be improved by adding a selection of confusable experts in the ensemble.

In sum, we believe the proposed approach to be usable in proofing tools.

2.7 Discussion

Concerning the generality of the approach, the proposed method applies to confusable cases in which the writer is forced to choose among a set of paradigmatic alternatives for derivation or inflection, and in which the alternatives have a similar

but discernable surface form that uniquely identifies at least one of the two alternatives. A word ending in *-dt* will almost exclusively be a present-tense singular verb form, or a typo of a word that should end in *-d*. This identification enables the knowledge-free extraction of these cases and their counterparts from large text corpora; no annotation effort is needed.

Intrinsic to our approach is the inclusion of "leaked" examples of words with other part-of-speech tags than just the main tag associated with the underlying inflection, e.g. in our case, past participles and nouns ending in *-d*. These words however can also be, and judging from web queries, are, frequently misspelled with *-dt* at the end. These occurrences will in principle also be corrected by our system.

Similar cases to which this method could be applied straightforwardly include confusable sets with two or more than two outcomes, as long as they keep the property of having marked alternative word endings of which at least one uniquely identifies a morphological inflection or derivation. If one is marked, then the counterparts can be extracted automatically as well. Some random examples that fit the bill would be Dutch diminutive inflection *-tje* vs. *-je*, *-etje*, *-pje*, and *-kje* (Daelemans et al. 1997a); English gerund/infinitive inflection *-ing* versus forms without *-ing*; *-y* versus *-ily*, and *-ation* versus *-ization*.

A final point concerns evaluation. On the larger issue of spelling correction, it has been argued that the precision and recall of error detection and correction (which are not the same, but in our case of two-way confusable outcomes, are conflated) constitute the best evaluation of a spelling correction system (Reynaert 2005). A spelling error detector should not produce false hits (detect an error when there is none) nor produce misses (fail to detect an error where there is one). Precision and recall are the appropriate measurements for these two aspects. However, to estimate precision and recall reliably, the test would involve processing a (preferably large) corpus of free text in which all actual errors are detected in advance. As these corpora are not available as yet, we have to downgrade our evaluation to recall estimates such as the one presented in this paper, in which a precompiled list of errors in context is presented to the error detector. We do not know yet the precision of our systems, i.e., the relative amount of cases it disagrees with a wordform in free text in which it actually flags an error correctly. Estimates on professionally written, edited text such as the text we train our classifiers on, indicate that this precision is only between 2% and 10% – but this is double-checked text that is supposed to be devoid of errors already. If in journalistic texts only 1 in 2000 words are incorrect (Church and Gale 1991), then a 99% correct classifier would raise about 20 alarms in a 2000-word text, while there would only be one error in the text; the precision could in that case be 5% at best.

In general we recommend that our method be applied to text that is written outside of professional context, such as a lot of the material on the world-wide web (Ringlstetter et al. 2006).

References

Banko, M. and E. Brill (2001), Scaling to very very large corpora for natural language disambiguation, *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 26–33.

Biemann, Christian (2007), *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*, PhD thesis, Leipzig University.

Church, Kenneth Ward and William A. Gale (1991), Probability scoring for spelling correction, *Statistics and Computing* **1** (2), pp. 93–103.

Daelemans, W., A. Van den Bosch, and A. Weijters (1997a), IGTree: using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11**, pp. 407–423.

Daelemans, W., P. Berck, and S. Gillis (1997b), Data mining as a method for linguistic analysis: Dutch diminutives, *Folia Linguistica* **XXXI** (1–2), pp. 57–75.

Even-Zohar, Y. and D. Roth (2000), A classification approach to word prediction, *Proceedings of the First North-American Conference on Computational Linguistics*, ACL, New Brunswick, NJ, pp. 124–131.

Geerts, G., W. Haeseryn, J. de Rooij, and M. van der Toorn (1984), *Algemene Nederlandse Spraakkunst*, Wolters-Noordhoff, Groningen and Wolters, Leuven.

Golding, A. R. (1995), A Bayesian hybrid method for context-sensitive spelling correction, *Proceedings of the 3rd workshop on very large corpora, ACL-95*.

Golding, A.R. and D. Roth (1999), A Winnow-Based Approach to Context-Sensitive Spelling Correction, *Machine Learning* **34** (1–3), pp. 107–130, Kluwer Academic.

Huang, J. H. and D. W. Powers (2001), Large scale experiments on correction of confused words, *Australasian Computer Science Conference Proceedings*, Bond University, Queensland AU, pp. 77–82.

Mangu, L. and E. Brill (1997), Automatic rule acquisition for spelling correction, *Proceedings of the International Conference on Machine Learning*, pp. 187–194.

Mooney, R. J. and M. E. Califf (1995), Induction of first-order decision lists: Results on learning the past tense of English verbs, *Journal of Artificial Intelligence Research* **3**, pp. 1–24.

Quinlan, J.R. (1993), C4.5*: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Reynaert, M. (2005), *Text-induced spelling correction*, PhD thesis, Tilburg University.

Ringlstetter, C., K. Schultz, and S. Mihov (2006), Orthographic errors in web pages: Toward cleaner web corpora, *Computational Linguistics* **32**

(3), pp. 295–340.

Sandra, D., F. Daems, and S. Frisson (2001), Zo helder en toch zoveel fouten! wat leren we uit psycholinguïstisch onderzoek naar werkwoordfouten bij ervaren spellers?, *Tijdschrift van de Vereniging voor het Onderwijs in het Nederlands* **30** (3), pp. 3–20.

Schone, P. and D. Jurafsky (2001), Knowledge-free induction of inflectional morphologies.

Van den Bosch, A. (2006), Scalable classification-based word prediction and confusible correction, *Traitement Automatique des Langues* **46** (2), pp. 39–63.

Van den Bosch, A. and W. Daelemans (2007), Dat gebeurd mei niet: Computationele modellen voor verwarbare homofonen., Academia Press.

Wu, D., Z. Sui, and J. Zhao (1999), An information-based method for selecting feature types for word prediction, *Proceedings of the Sixth European Conference on Speech Communication and Technology, EUROSPEECH'99*, Budapest.

Yarowsky, D. (1994), Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French, *Proceedings of the Annual Meeting of the ACL*, pp. 88–95.

Yarowsky, D. and R. Wicentowski (2000), Minimally supervised morphological analysis by multimodal alignment, *Proceedings of ACL-2000*, Morgan Kaufmann, San Francisco, CA, pp. 207–216.

3

Aligning linguistically motivated phrases

Lieve Macken and Walter Daelemans
Ghent University College and University of Antwerp

Abstract

In this paper, we describe the architecture of a sub-sentential alignment system that links linguistically motivated phrases in parallel texts.

We conceive our sub-sentential aligner as a cascade model consisting of two phases. In the first phase, anchor chunks are linked on the basis of lexical correspondences and syntactic similarity. In the second phase, we will focus on the more complex translational correspondences based on observed translation shift patterns. The anchor chunks of the first phase will be used to limit the search space in the second phase.

We present the first results of our sub-sentential alignment system, which links linguistically motivated chunks. In our baseline system, the obtained recall scores range from 44% to 59% and precision scores from 90% to 98% depending on text type.

We experimented with two different types of bilingual dictionaries to generate the lexical correspondences: a handcrafted bilingual dictionary and probabilistic bilingual dictionaries. We demonstrate that although the handcrafted dictionary is twice the size of the probabilistic dictionary, the obtained recall scores are lower.

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 37–52
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.

3.1 Introduction

Sub-sentential alignments are used among other things to create phrase tables for statistical phrase-based machine translation (SMT) systems. In existing SMT systems, a phrase is not linguistically motivated. It can be any contiguous sequence of words. There is a strong intuition that the use of linguistically relevant phrases can improve the performance of phrase-based SMT systems. The experiments by Groves and Way (2006) for French-English confirm this assumption.

A stand-alone sub-sentential alignment module however, is also useful for human translators if incorporated in CAT-tools, e.g. sophisticated bilingual concordance systems, in sub-sentential translation memory systems (Gotti et al. 2005), or for bilingual terminology extraction (Macken et al. 2008).

Several researchers demonstrated that the addition of linguistic information can improve statistically-based word alignment systems. Tiedemann (2003) for example combines association measures with additional linguistic heuristics based on part-of-speech, phrase type, and string similarity measures. While Tiedemann makes use of chunk information, the alignment process remains word-based. In our approach, the whole alignment process is primarily chunk-driven.

We conceive our sub-sentential aligner as a cascade model consisting of two phases. In the first phase *anchor chunks*, i.e. chunks that can be linked with a very high precision based on lexical correspondences and syntactic similarity are retrieved. In the second phase, we will focus on the more complex translational correspondences based on observed translation shift patterns. The anchor chunks of the first phase will be used to limit the search space in this second phase. This paper describes the first phase, namely the alignment of anchor chunks.

3.2 Architecture

The global architecture of our system is visualized in figure 3.1. The sub-sentential alignment system takes as input sentence-aligned texts, together with additional linguistic annotations (part-of-speech codes and lemmas) for the source and the target text.

Although the global architecture of our sub-sentential alignment system is language-independent, some language-specific resources are used. In the first phase, two *external* language-specific linguistic resources are needed: first, a bilingual lexicon to generate the lexical correspondences; second, tools to generate additional linguistic information (PoS tagger, lemmatizer and chunker).

We will simulate the different steps of the sub-sentential alignment process for the following sentence pair:

En: Madam President, last week's attacks on innocent civilians in New York and Washington shocked and outraged the civilised people across the world.

Nl: Mevrouw de Voorzitter, de aanvallen op onschuldige burgers die vorige week hebben plaatsgevonden in New York en Washington hebben de beschaafde volkeren in de gehele wereld geschokt en verontwaardigd.

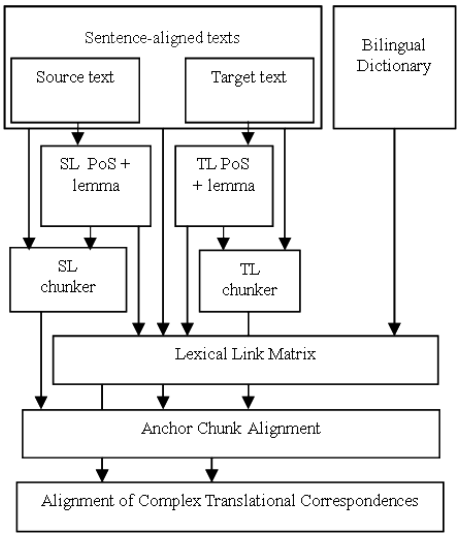

Figure 3.1: Outline of the full sub-sentential alignment system

In the first step of the process, the source and target sentences are divided into chunks based on PoS information, and lexical correspondences are retrieved from a bilingual dictionary. The chunk boundaries are visualized in figure 3.2 by means of horizontal and vertical lines. The lexical correspondences are marked by *x's*.

During anchor chunk alignment, the sub-sentential aligner links chunks based on lexical correspondences and chunk similarity. In figure 3.2, the anchor chunks are marked in light grey. In the example sentence, corresponding noun phrases, corresponding prepositional phrases and corresponding verb phrases are indicated.

In the second phase of the process, the sub-sentential aligner uses the anchor chunks of the first phase to retrieve the more complex translational correspondences. In figure 3.2, such a more complex translational correspondence is marked in dark grey. In the example sentence, the following translation shift can be observed: the English premodifier (*last week's*) is translated by a relative clause in Dutch (*die vorige week hebben plaatsgevonden*).

3.3 Bilingual lexicon

As explained in section 3.2, a bilingual dictionary is used in the first phase of the sub-sentential alignment process to indicate lexical correspondences in source and target sentences.

We experimented with two different types of bilingual dictionaries: a hand-crafted bilingual dictionary and probabilistic bilingual dictionaries, automatically extracted from bilingual corpora.

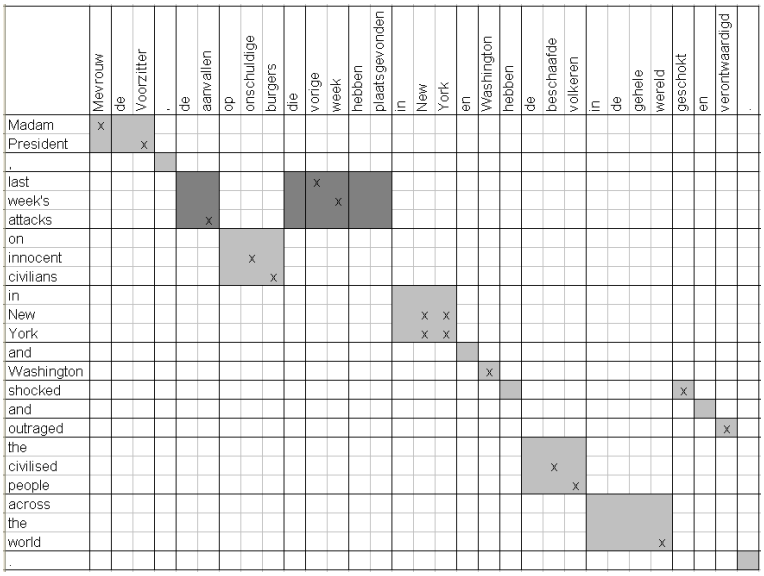

Figure 3.2: Simulation of the different alignment steps: chunk boundaries (horizontal and vertical lines), lexical correspondences (x's), anchor chunks (light grey) and complex translational correspondence (dark grey)

3.3.1 Handcrafted bilingual lexicon

The handcrafted bilingual lexicon was derived from the English-Dutch and Dutch-English Nl-Translex lexica[1](Goetschalckx et al. 2001). The Nl-Translex lexica contain apart from one-to-one correspondences (e.g. *force - kracht*) and compounds (e.g. *market sector - marktsegment*), also a large number of phrasal correspondences (e.g. *come into force - van kracht worden*, *agreement on government procurement - overeenkomst inzake overheidsopdrachten*).

As the major challenge of our research project is the automatic alignment of those more complex phrasal correspondences, we wanted to exclude these phrasal correspondences. However, as it was not possible to automatically distinguish between compounds and phrasal correspondences, we only retained all one-to-one correspondences.

The resulting bilingual dictionary contains 58,970 English-Dutch word pairs. More details can be found in table 3.1.

3.3.2 Probabilistic bilingual lexicon

We used a statistical word alignment package to derive a bilingual dictionary from a parallel corpus. Statistical word alignment is a semi-supervised method, which

[1]This work was carried out in the framework of the STEVIN DPC project.

Table 3.1: Nl-Translex English-Dutch dictionary containing only one-to-one correspondences

	# Entries
English-Dutch word pairs	58,970
English words	43,914
Dutch words	43,292

Table 3.2: Formal characterstics of the En-Nl parallel corpora used for dictionary creation (upper part) and of the resulting dictionary (lower part)

	2.4M	5.6M	9.3M
Aligned sentences	50,000	116,912	202,289
Total tokens	2,416,719	5,636,468	9,323,898
En word forms freq > 2	15,295	22,261	27,398
Nl word forms freq > 2	20,362	31,506	42,455
English-Dutch word pairs	16,728	21,486	28,342
English words	10,109	12,500	15,716
Dutch words	12,939	16,132	21,373

means that it starts from unannotated (raw) data. Most methods need a large sentence-aligned corpus to reliably estimate a statistical word alignment model. Statistical word alignment is based on the assumption of co-occurrence: words that are translations of each other co-occur more often than random in aligned sentence pairs. The output of a statistical word alignment model is a large bilingual word list with probability estimations.

As statistical word alignment tools need large sentence-aligned corpora, we opted for using the Europarl corpus (Koehn 2005). We created bilingual dictionaries using different parts of the Europarl corpus. The selected parts of the Europarl corpus were aligned on sentence level using the alignment tool that was released with the Europarl corpus. The corpora were tokenized and converted to lowercase.

Table 3.2 gives an overview of the formal characteristics of the resulting sentence-aligned parallel corpora that were used for deriving the bilingual dictionaries and the formal characteristics of the resulting bilingual dictionaries.
The most widely used statistical word alignment models are the IBM translation models (Brown et al. 1993). The most simple IBM model - IBM Translation Model One – is a purely *lexical* model: it only takes into account word frequencies in source and target sentences[2]. We used the Perl implementation of IBM Model

[2]The higher numbered IBM Models build on IBM Model One and take into account word order (distortion) and model the probability that a source word aligns to n target words (fertility).

One that is part of the Microsoft Bilingual Sentence Aligner (Moore 2002).

The IBM models allow only 1:n word mappings, and are therefore asymmetric. To overcome this problem, we ran the model in two directions: from English to Dutch and from Dutch to English. To get high-accuracy links, only the word pairs occurring in both the English-Dutch and Dutch-English word lists were retained, and the probabilities were averaged. To get rid of the noise produced by the translation model, only the entries with an averaged value of at least 0.1 were retained. This value was set experimentally.

To reduce the number of values, the averaged values were multiplied by 10 and only the integer part was retained. The obtained values allow us to rank the different translations according to frequency.

A sample of the resulting dictionary is shown in table 3.3. As the model was trained on a corpus of word forms, the dictionary does not abstract over word forms, e.g. *affordable - betaalbaar* and *affordable - betaalbare* are two separate entries in the dictionary. Model One can generate multiple translations for one word form, e.g. *affected* has three possible translations *getroffen, beïnvloed* and *getroffenen*, with *getroffen* being the most frequent translation.

Table 3.3: Sample of the English-Dutch probabilistic dictionary

English word form	Dutch word form	Frequency class
affect	invloed	1
affected	beïnvloed	1
affected	getroffen	4
affected	getroffenen	1
affection	genegenheid	1
afford	permitteren	3
afford	veroorloven	4
affordability	betaalbaarheid	2
affordable	betaalbaar	3
affordable	betaalbare	5

3.4 Additional linguistic annotations

Part-of-speech tagging and lemmatization for English was performed by the combined memory-based PoS tagger/lemmatizer, which is part of the MBSP tools (Daelemans and Van den Bosch 2005). Part-of-speech tagging and lemmatization for Dutch was performed by TADPOLE (Van den Bosch et al. 2007).

Although the MBSP toolkits contain chunking for English and Dutch, we opted for the development of two rule-based chunkers, the reason being that the English and Dutch shallow parsers adopt a different chunk definition. For example, adjacent verbs are clustered in one verbal group in the English memory-based shallow parser, but regarded as separate chunks in the Dutch memory-based shallow parser.

The rule-based chunkers for Dutch and English contain distituency rules. These rules add a chunk boundary when two consecutive part-of-speech codes cannot occur in the same constituent, e.g. between two finite verbs.

All Dutch and English texts of the test corpus (described in Section 3.6) were manually chunked, and the rule-based chunkers were evaluated by running the CoNLL-evalscript developed by Tjong Kim Sang, E.F. and Buchholz, S. (2000) on the test files. Precision scores of 93% (English) and 94% (Dutch) and recall scores of 95% (English and Dutch) were obtained.

3.5 Algorithm

As explained in section 3.2, we conceive our sub-sentential alignment system as a cascade model consisting of two phases. The objective of the first phase is to link *anchor chunks*, i.e. chunks that can be linked with a very high precision. Those anchor chunks are linked based on lexical clues and chunk similarity.

In order to link chunks based on lexical clues and chunk similarity, the following steps are taken for each sentence pair:

1. Creation of the lexical link matrix
2. Linking chunks based on lexical correspondences and chunk similarity
3. Linking adjacent function word chunks and final punctuation

The different steps are described in more detail below.

3.5.1 Creation of the lexical link matrix

Prior to creating the lexical link matrix, all possible translations for each word in the source and target sentence are retrieved from the bilingual dictionary. As explained in section 3.3, the probabilistic bilingual dictionary contains English-Dutch word pairs with numeric values that denote the frequency class of the word pair[3].

For each source and target word, all translations for the word form and the lemma are retrieved from the bilingual dictionary. If only the lemma of the source or target word is found in the bilingual dictionary, the resulting frequency weight is cut in half.

In the process of building the lexical link matrix, function words are neglected. Given the frequency of function words in a sentence, linking function words based on word alignment information alone often results in erroneous alignments. For that reason no lexical links are created for the following word classes: determiners, prepositions, coordinating conjunctions, possessive pronouns and punctuation symbols.

For all content words, if a source word occurs in the set of possible translations of a target word, or if a target word occurs in the set of possible translations of

[3]As in the Nl-Translex dictionary no frequency information is available, all word pairs get the same value.

the source words, a lexical link is created. Identical strings in source and target language are also linked.

The resulting lexical link matrix for our example is shown in figure 3.3.

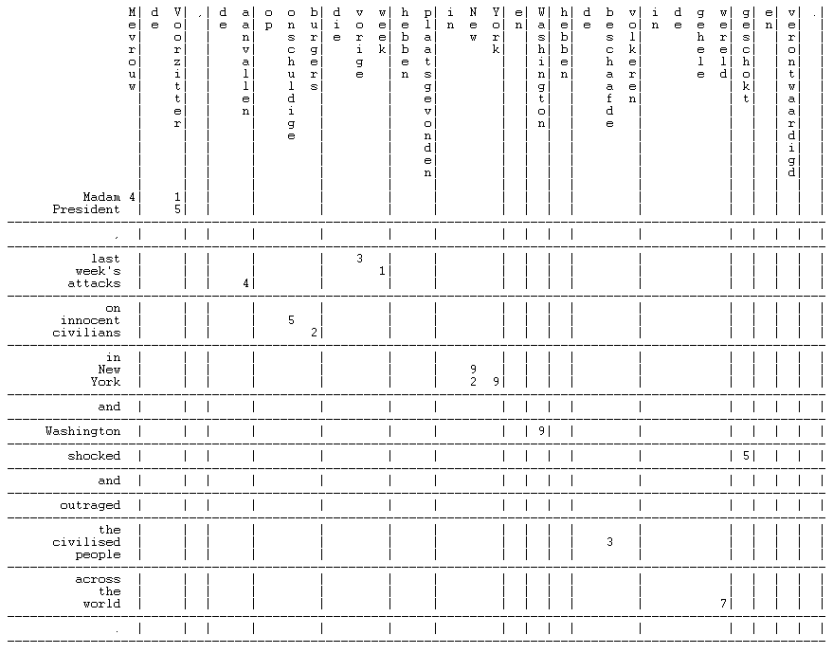

Figure 3.3: Lexical link matrix containing frequency weights

3.5.2 Linking anchor chunks

The problem of linking chunks based on lexical correspondences and similar chunks can be decomposed in two subproblems:

1. Selecting candidate anchor chunks
2. Testing chunk similarity of the candidate anchor chunks

Selecting candidate anchor chunks

The candidate anchor chunks are selected based on the information available in the lexical link matrix. For each source chunk a **contiguous** candidate target chunk is constructed. The contiguous candidate target chunk is built by concatenating all target chunks from a *begin index* until an *end index*. The begin index points to the first target chunk with a lexical link to the source chunk under consideration. The end index points to the last target chunk with a lexical link to the source chunk

under consideration. Possible intermediate chunks can contain additional lexical links, but this is not necessarily the case. If a source word contains more than one lexical link, the lexical link with the highest frequency weight is used.
In this way, the following contiguous 1:1 and 1:n candidate target chunks are built for our example:

Madam President	Mevrouw | de Voorzitter
last week's attacks	de aanvallen | op onschuldige burgers | die vorige week
on innocent civilians	op onschuldige burgers
in New York	in New York
Washington	Washington
shocked	geschokt
the civilised people	de beschaafde volkeren
across the world	in de gehele wereld

For some source chunks, it is also useful to build a **non-contiguous** candidate target chunk. The non-contiguous candidate target chunks are built by concatenating all target chunks with a lexical link to the source chunk under consideration. In our example, only one non-contiguous target chunk is constructed:

last week's attacks de aanvallen . . . vorige week

The process of selecting candidate chunks as described above, is performed twice: a first time starting from the source sentence; a second time starting from the target sentence. The second time, only those chunks for which no similarity test was performed are taken into consideration.

Testing chunk similarity

For each selected candidate pair, a *similarity test* is performed. Chunks are considered to be similar if at least a certain percentage of words of source and target chunk(s) are either linked by means of a lexical link or can be linked on the basis of corresponding part-of-speech codes.

All word classes can be linked based on PoS codes. In the candidate anchor chunk *the civilised people - de beschaafde volkeren*, one lexical clue (*civilised - beschaafde*) is sufficient to pass the similarity test as *the* and *de*, and *people* and *volkeren* are linked based on corresponding PoS codes.

The percentage of words that have to be linked was empirically set at 80% for contiguous chunks and 100% for non-contiguous chunks. The percentage of linked words is calculated as follows:

$$\frac{\text{\# words linked of source chunk + \# words linked of target chunk}}{\text{Total \# source chunk words + total \# target chunk words}}$$

The candidate anchor chunk *across the world - in de gehele wereld* contains one lexical link *world - wereld* and two PoS-links *across - in* and *the - de*. Hence the percentage of linked words $= (3 + 3)/(3 + 4) = 0.86$.

If a candidate anchor chunk passes the similarity test, the information in the matrix is updated as follows:

- All lexical links inside an anchor chunk are marked with the label **S** (Sure lexical links)
- Words linked based on corresponding part-of-speech codes are marked with the label **p** (PoS links)
- The label **r** is used to mark lexical links that are removed. If a source or target word had multiple lexical links, all lexical links other than the one(s) in the anchor chunk get the label **r**.

3.5.3 Linking adjacent function word chunks

In a final step, chunks consisting of one function word – mostly punctuation marks and conjunctions – can be linked based on corresponding part-of-speech codes if their left or right neighbours on the diagonal are anchor chunk. Corresponding final punctuation marks are also linked.

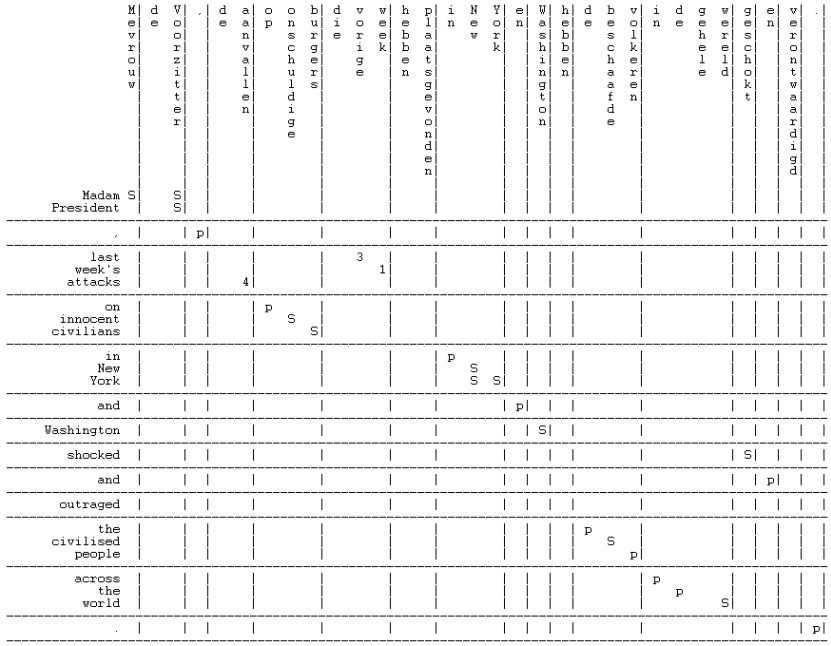

Figure 3.4: Matrix containing anchor chunks (**S** and **p** labels) and remaining lexical links with frequency weights

The resulting matrix for our example is shown in figure 3.4. In the example,

the following anchor chunks were retrieved:

Madam President	Mevrouw | de Voorzitter
,	,
on innocent civilians	op onschuldige burgers
in New York	in New York
and	en
Washington	Washington
shocked	geschokt
and	en
the civilised people	de beschaafde volkeren
across the world	in de gehele wereld

All retrieved anchor chunks but one can be considered to be entirely correct: *shocked* should have been linked to *hebben ... geschokt*, so the anchor chunk *shocked - geschokt* is only partially correct. In section 3.6, we describe how we evaluated the performance of the system.

3.6 Experimental results

A manual reference corpus was created that includes three different text types: user manuals, press releases and proceedings of plenary debates. Three different types of links were used: regular links for straightforward correspondences (e.g. *innocent - onschuldige*, *New York - New York*), fuzzy links for translation-specific shifts of various kinds (e.g. *last week's - die vorige week hebben plaatsgevonden*), and null links for words for which no correspondence could be indicated (deletions or additions). In the manual reference corpus, different units could be linked (words, word groups, paraphrased sections, punctuation). More details on the creation of the manual reference corpus can be found in (Macken 2007).

To evaluate the system's performance, the links created by the system were compared with the links in the manual reference files. Table 3.4 gives an overview of the number of words and documents used for testing the system.

To be able to compare the alignments of the system with the reference alignments, all phrase-to-phrase alignments were converted into word-to-word alignments by linking each word of the source phrase to each word of the target phrase

Table 3.4: En-Nl Test data

Text type	# Words	# Texts
Proceedings EP	3,139	7
Press Releases	4,926	4
User Manuals	4,010	2
Total	12,075	13

(all-pairs heuristic).

3.6.1 Metrics

The results of the experiments were evaluated in terms of precision and recall, which are widely used in the context of information retrieval. If we would calculate precision and recall on all word-to-word links, all links would be equally important. However, as Melamed (2001) pointed out, an evaluation metric that treats all links as equally important would place undue importance on words that were linked more than once (e.g. all word-to-word links resulting from the phrasal alignments). Therefore, a weight is assigned to each word-to-word link, and precision and recall are calculated on the weights of the word-to-word links.

We use the weighting method developed by Davis et al. (2007), which is a refinement of the weighting principles introduced by Melamed (2001). In this weighting scheme, every word contributes 0.5 to the total weight. In case of interlinked word-to-word links from the phrasal alignments, each link is assigned the total weight of the phrasal alignment divided by the number of word-to-word links. In the example of *last week's - die vorige week hebben plaatsgevonden*, the total weight of the phrasal alignment is 3.5 ($(2+5) \times 0.5$), and each word-to-word link gets a weight of 0.35 ($3.5/(2 \times 5)$). In the case of a regular word-to-word link (e.g. *innocent - onschuldige*), the resulting weight of the word-to-word link is 1 ($((1+1) \times 0.5)/(1 \times 1)$).

Precision and recall are then calculated on the weights, by using the following equations:

$$Precision = \frac{\text{total system weight of corresponding word-to-word links}}{\text{total system weight}}$$

$$Recall = \frac{\text{total reference weight of corresponding word-to-word links}}{\text{total reference weight}}$$

3.6.2 Results

The results presented here are the results of a system that uses the probabilistic dictionary trained on the 9.3M word corpus.

We considered all word-to-word links at two different stages in the system. A first time after dictionary lookup, and a second time after the alignment of chunks based on syntactic similarity. However, as it is also interesting to assess the reliability of the alignments of the anchor chunks, precision and recall are also calculated on only the word-to-word links that are part of the aligned anchor chunks. Table 3.5 contains the results per text type for all the texts of the test corpus.

In the columns under the heading *DCT*, the results after dictionary lookup are displayed. It it worth noticing that although the bilingual lexicon was trained on Europarl data, the coverage is quite good on other domains. The high figure for Press Releases can be explained by the high number of technical terms that are

Table 3.5: Normalized precision and recall on all word-to-word links at different stages in the system per text type

	DCT		DCT + AC		AC	
	Prec	Rec	Prec	Rec	Prec	Rec
Proceedings EP	.78	.28	.90	.44	.92	.38
Press Releases	.90	.34	.98	.59	.98	.54
User Manuals	.85	.27	.95	.46	.97	.40

identical in source and target text (the Press Releases test corpus contained 12% identical strings).

The columns under the heading *DCT + AC* contain the results after the alignment of syntactically similar chunks. During the alignment process, extra word-to-word links can be created for words belonging to the anchor chunks based on corresponding PoS codes (e.g. function words, adjectives). This explains the increase in recall. On the other hand, some disambiguation takes place for words that were linked several times. This explains the increase in precision.

In the last columns under the heading *AC*, precision/recall results are given for only the word-to-word links belonging to the chunks that were aligned based on syntactic similarity. The obtained precision scores (between .92 and .98) seem high enough to use the aligned chunks as anchors in the second phase of the alignment process.

We also investigated the impact of the size of the training corpus used for dictionary creation on the test results. We compared the obtained precision and recall scores at the different stages in the system on all the test files. As can be seen in figure 3.5, the size of the training corpus – and hence the size of the resulting dictionary – has a positive impact on recall at all stages in the system. No difference in precision was observed.

It is also interesting to compare the performance of the handcrafted bilingual lexicon with the performance of the probabilistic bilingual lexicon. Although the Nl-Translex dictionary is twice the size of the probabilistic dictionary trained on the 9.3M word corpus, the obtained recall scores are lower at all stages of the system[4]. The alignments retrieved by the system using the Nl-Translex system are more precise after dictionary lookup. But no difference in precision is observed if we only take into account the retrieved anchor chunks.

As explained in section 3.3.2, the probabilistic dictionaries were automatically extracted from a corpus containing word forms. We examined the impact of lemmatizing the training corpus prior to dictionary creation on the test results. By lemmatizing the training corpus, we expect that abstracting over word forms will

[4]It is also worthwhile to mention that the Nl-Translex dictionary and the probabilistic dictionary (trained on the lemmatized corpus) contain different word pairs: only 21% of the entries of the probabilistic dictionary are part of the NL-Translex dictionary.

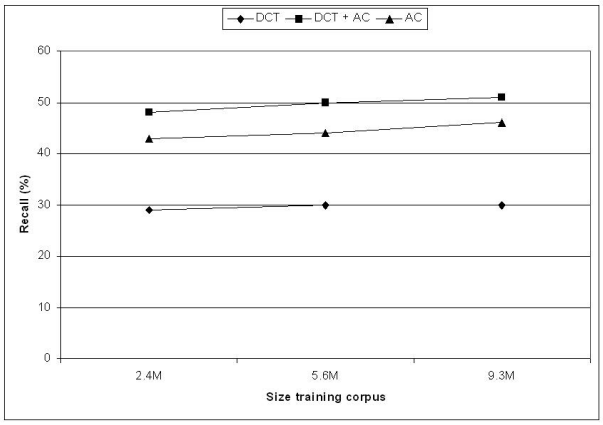

Figure 3.5: Impact of size training corpus on recall

increase the overall recall scores.

As the size of the dictionary might influence the impact of lemmatization, we trained probabilistic dictionaries on four different corpora: the lemmatized and word form version of both the 2.4M and the 9.3M word corpus respectively[5].

Lemmatization has a positive impact on recall for the system using the 2.4M word corpus. But the recall improvement is less clear for the system using the 9.3M word corpus. The reason being that the coverage of the 9.3M word forms dictionary is quite high[6]. The precision scores are slightly better for the word forms corpora.

3.7 Conclusion

In this paper, we have described the global architecture of our sub-sentential alignment system. We conceive our sub-sentential aligner as a cascade model with two phases. In the first phase *anchor chunks*, i.e. chunks that can be linked with a very high precision based on lexical correspondences and syntactic similarity are retrieved. In the second phase, we will focus on the more complex translational correspondences based on observed translation shift patterns. The anchor chunks of the first phase will be used to limit the search space in the second phase.

The objective of the first phase was to link *anchor chunks*, i.e. chunks that can be linked with a very high precision. In our baseline system, on average 40-50% of the words can be linked with a precision ranging from 90% to 98%. The obtained

[5]In the lemmatized systems, the retrieval of the lemmatized form is not penalized by reducing the frequency weight.

[6]We lemmatized the resulting dictionary extracted from the 9.3M word forms corpus off-line and compared it with the dictionary extracted from the lemmatized 9.3M corpus. An overlap of 95% was obtained. The overlap on the resulting dictionaries trained on the 2.3M word corpus was 85%

precision scores seem high enough to use the aligned chunks as anchors in the second phase of the alignment process.

We experimented with two different types of bilingual dictionaries to generate the lexical correspondences: a handcrafted bilingual dictionary and probabilistic bilingual dictionaries. We demonstrate that although the handcrafted dictionary is twice the size of the probabilistic dictionary, the obtained recall scores are lower. No difference in precision is observed for the retrieved anchor chunks.

We demonstrated that lemmatizing the training corpus prior to dictionary extraction can increase recall for small training corpora. As expected, increasing the size of the training corpora has a positive impact on the overall recall scores.

References

Brown, P.F., V.J. Della Pietra, S.A. Della Pietra, and R.L. Mercer (1993), The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics* **19** (2), pp. 263–311.

Daelemans, W. and A. Van den Bosch (2005), Application to shallow parsing, *in* Daelemans, Walter and Antal Van den Bosch, editors, *Memory-based language processing*, Cambridge University Press, Cambridge, United Kingdom, pp. 85–103.

Davis, P.C., Z. Xie, and K. Small (2007), All Links are not the Same: Evaluating Word Alignments for Statistical Machine Translation, *in* Maegaard, Bente, editor, *Machine Translation Summit XI*, European Associaton for Machine Translation, Copenhagen, Denmark, pp. 119–126.

Goetschalckx, J., C. Cucchiarini, and J. Van Hoorde (2001), Machine Translation for Dutch: the NL-Translex Project.

Gotti, F., P. Langlais, E. Macklovitch, D. Bourigault, B. Robichaud, and C. Coulombe (2005), 3GTM: a third-generation translation memory, *3rd Computational Linguistics in the North-East (CLiNE) Workshop*, Gatineau, Québec.

Groves, D. and A. Way (2006), Hybridity in MT: Experiments on the Europarl corpus, *11th Conference of the European Association for Machine Translation*, Oslo, Norway.

Koehn, P. (2005), Europarl: a parallel corpus for statistical machine translation, *Tenth Machine Translation Summit*, Phuket, Thailand, pp. 79–86.

Macken, L. (2007), Analysis of translational correspondence in view of sub-sentential alignment, *METIS-II Workshop on New Approaches to Machine Translation*, Leuven, Belgium, pp. 97–105.

Macken, L., E. Lefever, and V. Hoste (2008), Linguistically-based sub-sentential alignment for terminology extraction from a bilingual automotive corpus, *22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, United Kingdom.

Melamed, D.I. (2001), Manual annotation of translational equivalence, *in*

Melamed, Dan I., editor, *Empirical methods for exploiting parallel texts*, MIT Press, Cambridge, Massachusetts, pp. 65–77.

Moore, R.C. (2002), Fast and accurate sentence alignment of bilingual corpora, *5th Conference of the Association for Machine Translation in the Americas*, Machine Translation: from research to real users, Tiburon, California, pp. 135–244.

Tiedemann, J. (2003), Combining Clues for Word Alignment, *10th Conference of the European Chapter of the ACL (EACL03)*, Budapest, Hungary.

Tjong Kim Sang, E.F. and Buchholz, S. (2000), Introduction to the CoNLL-2000 Shared Task: Chunking, *CoNLL-2000 and LLL-2000*, Lisbon, Portugal, pp. 127–132.

Van den Bosch, A., B. Busser, W. Daelemans, and S. Canisius (2007), An efficient memory-based morphosyntactic tagger and parser for Dutch, *Computational Linguistics in the Netherlands 2006*, Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting, Leuven, Belgium, pp. 191–206.

4

Features for automatic discourse analysis of paragraphs

Daphne Theijssen, Hans van Halteren, Suzan Verberne and Lou Boves
Department of Linguistics, Radboud University Nijmegen

Abstract

In this paper, we investigate which information is useful for the detection of rhetorical (RST) relations between (Multi-) Sentential Discourse Units ((M-)SDUs) – text spans consisting of one or more sentences – within the same paragraph. In order to do so, we simplified the task of discourse parsing to a decision problem in which we decided whether an (M-)SDU is rhetorically related to either a preceding or a following (M-)SDU. Employing the RST Treebank (Carlson et al. 2003), we offered this choice to machine learning algorithms together with syntactic, lexical, referential, discourse and surface features. Next, we determined which of the features were most useful for predicting the direction of the relation by ranking them on the basis of three different metrics. Highly ranked features that predict the presence of a rhetorical relation are syntactic similarity, word overlap, word similarity, continuous punctuation and many reference features. Other highly ranked features predict the absence of a relations (i.e. are used to introduce new topics or arguments): time references, proper nouns, definite articles, the word *further* and the verb *bring*.

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 53–68
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: d.theijssen@let.ru.nl

4.1 Introduction

In the field of language and speech technology, the analysis of discourse structures in texts receives much attention. A commonly used model for discourse analysis is Rhetorical Structure Theory (RST), which was developed by Mann and Thompson (1988). RST is based on the idea that rhetorical relations exist between adjacent spans of text, of which one span, called the NUCLEUS, is more important for the purpose of the author than the other spans, called the SATELLITES. Sometimes spans are equally vital; then the relation is called multi-nuclear. The smallest text spans that can hold rhetorical relations are called Elementary Discourse Units (EDUs). The popularity of RST has led to the development of an RST Treebank of manually annotated English texts, which is available for training and testing purposes (Carlson et al. 2003). It consists of 385 Wall Street Journal articles from the Penn Treebank (Marcus et al. 1993) with a total of 176,383 words. In the RST Treebank, mono-nuclear relations are always binary (53 of 78 relation types), while multi-nuclear relations can also occur between more than two spans (the remaining 25 relation types).

The literature shows that various automatic RST parsers have been created. A state-of-the-art and publicly available system for automatic RST parsing of English texts is Sentence-level PArsing of DiscoursE (SPADE, Soricut and Marcu 2003). It produces an RST tree for every sentence in the input, but makes no attempt to find relations between sentences and at higher levels. In previous research by Marcu (Marcu 1999, Marcu 2000) and by others, the extraction of rhetorical relations at all text levels has been addressed. It has resulted in the discourse parsers RASTA (Rhetorical Structure Theory Analyzer, Corston-Oliver 1998) and DAS (Discourse Analyzing System, LeThanh 2004). However, RASTA nor DAS is generally available, nor is any of the systems reported by Marcu.

Because no system for automatic discourse (RST) analysis that is suitable for text analysis at all text levels is available, we decided to start the development of one. SPADE provides a first step towards such a system by splitting sentences into EDUs and providing RST trees for each sentence. A second step could be to find relations between text spans consisting of at least one sentence within the same paragraph. The goal of this paper is to investigate which information about the sentences may be useful for this second step. In other words, we attempt to answer the question: "Which features are most effective/powerful for predicting the presence of rhetorical (RST) relations between (Multi-)Sentential Discourse Units within paragraphs in English?" We introduce the term *(Multi-)Sentential Discourse Unit* ((M-)SDU) as a text span with a length of at least one sentence and at most one paragraph, forming a discourse unit in a text.

Following Soricut and Marcu (2003), we limited ourselves to binary relations[1]. We reduced discourse analysis to a classification task that we offered to various machine learning algorithms together with an inventory of potentially relevant features. Next, we ranked the features with the help of the classification algorithms

[1] In the RST Treebank, 99% of the rhetorical relations are binary (the remaining 1% being multi-nuclear relations between more than two spans).

and a feature ranking metric.

The organization of this paper is as follows: In Section 4.2, we introduce the classification task, describe the potentially relevant features and present the accuracies reached by the classification algorithms. The ranking of the features is described in Section 4.3. The final Section (8.4) contains our overall conclusion and gives recommendations for future research.

4.2 Discourse analysis as a classification task

In order to determine which features may be relevant for automatic discourse analysis of paragraphs, we simplified the problem of RST discourse analysis to a task that can easily be performed by machine learning algorithms. We ignored the type and direction of the rhetorical relations to prevent data sparseness.

In a binary RST tree, we considered each *triple* of three adjacent (M-)SDUs $x - y - z$ in the same paragraph in which either $x - y$ or $y - z$ are rhetorically related. In such triples, two (M-)SDUs are fixed, namely those between which the rhetorical relation holds. The third one is only restricted by three conditions: (1) it should be an (M-)SDU, thus represented by a separate node in the RST tree, (2) it should be adjacent to one of the two fixed (M-)SDUs, and (3) it should be in the same paragraph. The same relation can thus be present in more than one triple in our data set.

We employed classification algorithms to classify the triples according to the position of the relation in the triple: on the *left* ($x - y$) or on the *right* ($y - z$). Note that x, y and z may consist of more than one sentence and may thus contain relations between (M-)SDUs themselves. This means that our approach covers all relations between (M-)SDUs in the paragraph, as the eventual object of discourse parsing of paragraphs requires.

With the help of a Perl script, we automatically extracted 2136 triples (1196 *right*, 940 *left*) from 942 different paragraphs in 246 different Wall Street Journal texts in the RST Treebank.

4.2.1 Features

Machine learning algorithms need information about the triples to be able to classify them. We followed two strategies to establish an inventory of potentially relevant features: (1) by considering the literature on approaches previously taken by Corston-Oliver (1998), Marcu (1999), Marcu (2000) and LeThanh (2004), and (2) by studying a sample of the RST Corpus[2]. The result is a list of features that we subdivided into surface features, syntactic features, lexical features, reference features and discourse features. All features values were determined automatically with the help of a Perl script.

[2]We randomly selected 30 texts with a length of at least 5 sentences in the RST Treebank. Next, we extracted all binary relations between (M-)SDUs within the same paragraph. We established the proportion of SDU – SDU, SDU – M-SDU, M-SDU – SDU and M-SDU – M-SDU relations in the treebank and randomly selected 200 relations with the same proportions.

Surface features

Marcu (1999) used the presence of words and part-of-speech (POS) tags as features in his machine learning approach. We included all lemmas and POS tags present in the data. For the purpose of lemmatization we employed the CELEX lexicon (Baayen et al. 1995), and we took the Part-of-Speech tags from the Penn Treebank. We also used trigrams containing either the word token or the POS tag in each slot (three adjacent words are thus represented by eight different triples). The (M-)SDU lengths (in sentences and in words) were taken into account as well.

Since each lemma, POS tag and trigram was considered a separate feature, the number of surface features was too large (over 18,000) to be computationally feasible. We therefore chose the 1,000 most useful surface features according to the feature selection algorithm Relief (Kononenko 1994). Only these features have been applied in the experiments and analyses[3].

Syntactic features

In Corston-Oliver (1998), the syntactic features *tense* (e.g. past), *aspect* (e.g. progressive) and *polarity* (e.g. negative) are introduced. We have used similar information by counting the (relative) number of modals, infinitives, gerunds, past forms and present forms in each (M-)SDU, and by checking the clauses for negation.

A potentially relevant feature we discovered in the sample of the RST Treebank is *syntactic similarity*, as exemplified in Table 4.1. Existing metrics to establish syntactic similarity were not suitable for our purpose: parser evaluation metrics such as *Parseval* (Black et al. 1991) require that the two compared structures describe the same sentence, and methods such as *document fingerprinting* (Bernstein and Zobel 2005) establish the similarity of larger texts, not of small units such as (M-)SDUs. We have developed a simple metric which determines the syntactic similarity of two (M-)SDUs by comparing their clause structures (Theijssen 2007). The result is a continuous value between 0 and 1.

Lexical features

The example illustrating syntactic similarity also indicated the relevance of cue phrases such as *but*, *for this reason*, *in short*, etc. This has also been argued by Corston-Oliver (1998), Marcu (1999), Marcu (2000) and LeThanh (2004). We have included all 207 cue phrases that LeThanh considers relevant above clause level[4]. LeThanh (2004) also introduced noun phrase (NP) and verb phrase (VP) cues such as *goal* (NP), *purpose* (NP and VP) and *result from* (VP). We included all her 41 NP and 56 VP cues in our experiments[5].

[3]This selection was done separately for each individual training set in the ten-fold cross-validation described in Section 4.2.2. In total, 7,828 unique surface features have been selected.

[4]Only 21 of them were found in our total data set of 2136 triples.

[5]In our total data set of 2136 triples, 20 NP and 43 VP cues were present.

Table 4.1: Example of syntactic similarity in wsj_0688

	SDU 1	SDU 2
adverb	-	*But*
PP	*For instance*	*on the West Coast, where profitable oil production is more likely than in the midcontinent region, the Bakersfield, Calif.*
NP subject	*employment in Denver*	*office staff of 130*
modal	*will*	*will*
lexical verb	*be reduced*	*grow*
PP	*to 105*	*by 175*
PP	*from 430*	*to 305*

Other lexical features we found in the literature and the data sample were *word overlap* and *word similarity*. We defined three types of word overlap, namely the relative number of overlapping tokens, lemmas and stems. Word similarity was measured by employing Extended Gloss Overlap in *WordNet::Similarity* (Pedersen et al. 2004) and by consulting Lin's (1998) *Dependency Thesaurus*. In the example below, similar words are marked.

> *The FDA has said it **presented evidence** it uncovered to the company indicating that Bolar substituted the brand-name product for its own to gain government approval to sell generic versions of Macrodantin. Bolar has **denied** that it switched the brand-name product for its own in such testing.*
> — wsj_2382

Seeing data instances such as that below, we expected that the *presence of time references* could also be a relevant feature:

> *Until recently, Adobe had a lock on the market for image software, but last month Apple, Adobe's biggest customer, and Microsoft rebelled. Now the two firms are collaborating on an alternative to Adobe's approach, and analysts say they are likely to carry IBM, the biggest seller of personal computers, along with them.*
> — wsj_2365

Reference features

We found that many of the rhetorically related (M-)SDUs in the sample of the RST Treebank contained references. Referring to previously mentioned items by using personal pronouns, definite articles, demonstrative pronouns and (wh-)determiners (e.g. *which*) was therefore represented in features indicating their presence and their relative frequency in the (M-)SDU. We also established a list of 31 reference adverbs and adjectives (e.g. *other*) that we included in our approach. The list was

based on the words found in the sample, supplemented with synonyms taken from the thesaurus of Microsoft Word 2003[6].

Corston-Oliver's (1998) system also includes an anaphora resolver which automatically finds the antecedents of reference words. Since the system is not generally available, we employed the anaphora resolution tool GuiTAR (Poesio and Alexandrov-Kabadjov 2004) to check whether an anaphoric relation was present between two (M-)SDUs.

We here introduce a new feature *NP simplification*, being the lack of NP modifiers or NP heads in noun phrases that have been used previously in the text. Both types are illustrated below: the head *transaction(s)* in the first example, and the modifiers *Wall Street Journal's "American Way of Buying"* in the second example are missing in the second underlined phrase:

> *Grimm counted 16 transactions valued at $1 billion or more in the latest period, twice as many as a year earlier. The largest was the $12 billion merger creating Bristol-Myers Squibb Co.*
> — wsj_0645

> *When consumers have so many choices, brand loyalty is much harder to maintain. The Wall Street Journal's "American Way of Buying" survey found that 53% of today's car buyers tend to switch brands. For the survey, Peter D. Hart Research Associates and the Roper Organization each asked about 2,000 U.S. consumers about their buying habits.*
> — wsj_1377

Discourse features

The last type of features concerns information on the structure of the text. From what we saw in the sample of the RST Treebank, we expected that the presence of continuous punctuation is a helpful cue for the detection of rhetorical relations. In the example below, the second quotation part consists of more than one sentence. Moreover, both sentences are between (the same) parentheses:

> *("A turban," she specifies, "though it wasn't the time for that 14 years ago. But I loved turbans.")*
> — wsj_1367

Also, we included information on the position of the (M-)SDU in the text (paragraph number) and in the paragraph (sentence number) as a discrete feature. The internal (binary) discourse structure of the (M-)SDU was also taken into account. We represented this by the number of EDUs and the nuclearity (NUCLEUS or SATELLITE) of both spans in the highest rhetorical relation. For example, if the internal discourse structure of an (M-)SDU is N1-S3, it contains a relation between a NUCLEUS span of 1 EDU and a SATELLITE span of 3 EDUs.

[6]The English thesaurus of Microsoft Word 2003 was developed for Microsoft by Bloomsbury Publ.

4.2.2 Method

We have formulated definitions for each of the features and have written Perl scripts for the automatic extraction of the feature values. Where possible, we used existing resources and tools, e.g. the syntactic analyses in the Penn Treebank. Depending on the form of the feature, its value had to be extracted for each (M-)SDU *x*, y and *z* in the triple, or for both pairs *x*-*y* and *y*-*z*. In total, 1,836 features were used, being the 1,000 best surface features, 20 syntactic features, 718 lexical features, 84 reference features and 14 discourse features. For details on the definition and extraction of the features, the reader is referred to Theijssen (2007).

We applied five machine learning algorithms: *Naive Bayes*, *k-Nearest Neighbours* (*kNN*), *Support Vector Machines* (*SVM*), *Decision Trees* and *Maximum Entropy*. The first four are present in the Orange software (Demsar et al. 2004) and we chose to employ those implementations. For Maximum Entropy we used the implementation of Zhang (2004). Since there was not enough data to establish the optimal parameters for each algorithm, we applied the algorithms with their default settings. The continuous features were made discrete by dividing their range into seven equal-frequency intervals with the 'discretization'-function in Orange, and were offered in this form to Naive Bayes and Maximum Entropy. Although potentially important knowledge about the exact distribution of the feature values is lost in this way, we believe it is a necessary step to reach (interpretable) output.

Due to the rather small number of triples, we decided to apply ten-fold cross-validation on all cases. It would not be fair to place some triples extracted from a particular Wall Street Journal text in the train data and other triples extracted from the same text in the test data. Therefore we had to manually split the data into partitions with equal numbers of triples and of texts.
The results are compared to the accuracy reached by chance: 56.0% (always choosing *right*). In order to establish an upper bound of the classification task, we randomly selected 50 triples in 50 different paragraphs. Two human analyzers who are familiar with RST reached an average accuracy of 87.0% on the classification task. Their inter-annotator agreement was substantial: Cohen's (1960) Kappa = 0.78.

4.2.3 Results

Since the machine learning task concerns choosing between only two classes (*left* and *right*), and the distribution of both classes is known, the machine learning results are represented by the *accuracy*, being the number of correctly classified cases in the test set divided by the total number of cases in the test set. The accuracies reached by the algorithms can be found in Table 4.2. Comparison with existing discourse parsers is not possible because they perform a different task.

Only Naive Bayes and Maximum Entropy reached an accuracy that is significantly better than chance. To check whether the other algorithms were affected by the large number of features and the low number of cases, we offered fewer features to them by employing Relief for feature selection, and selecting the best fea-

tures for each partition. As expected, the performance of kNN, SVM and Decision Trees increased when fewer features were offered, but only SVM ever performed significantly better than chance[7]. Naive Bayes and Maximum Entropy showed the opposite effect: their classification accuracy decreased when using fewer features.

Table 4.2: Accuracies reached. * means $p<0.001$ (compared to chance)

Chance	Naive Bayes	kNN	SVM	DecTrees	MaxEnt	Human
56.0%	60.0%*	51.1%	56.9%	53.1%	60.9%*	87.0%*

4.2.4 Discussion

Despite our efforts to include good representations of all potentially relevant information, the accuracies reached by the machine learning algorithms were only slightly better than chance (56.0%). An explanation for the results could be that the default settings in Orange were not optimal for the given task and data. The default k in kNN, for example, is the square root of the number of cases in the training set. It is possible that a lower k could increase the accuracy reached and thereby the suitability of the system and its model. Adjusting the parameter setting is thus highly recommended for future research.
Since it is not our goal to reach high accuracy on the classification task, but to establish what information (which features) are useful in the detection of rhetorical relations, the problem is less severe than it seems. Still, an important consequence of the low accuracies reached by these classification algorithms is that analyzing the models is speculative and should thus be performed with care.

4.3 Feature ranking

In order to discover which of the features in our feature set are most useful, we ranked them on the basis of three different metrics: (1) a metric based on the model of Naive Bayes, (2) a metric based on the model of Maximum Entropy, and (3) a feature ranking metric developed by one of the authors (Van Halteren, see below). Earlier in our research we employed the feature selection algorithm that comes standard with Orange (Demsar et al. 2004), being Relief (Kononenko 1994). Since Relief shows only a low to medium correlation with the other three metrics (the Spearman's rank correlation coefficient is 0.33 to 0.51 with $p<0.001$), we excluded it from the ranking described in this section. Exploring the differences between Relief and the other three metrics is beyond the scope of this paper.

[7] When provided with the best 100 features, the accuracy is 58.7% with chi-square 6.17, $p<0.05$

4.3.1 Method

Given the significant improvement over chance, we believe Naive Bayes and Maximum Entropy were able to sift the information from the sets of features with some success. Assuming that this sifting is expressed in the model parameters, we attempted to extract an indication of feature importance. As for the systems that were not able to perform better than chance, they were obviously unable to discover the information and any ranking is not likely to provide a useful measurement of feature importance.

To find a relevance score for the features following the model of Naive Bayes, we established the probability of each feature given the class. We approached this by considering both classes *left* and *right* and counting the number of times a certain feature value occurred with that class, and divided it by the total number of cases with the class in the training set. We then looped through all cases in the test set and divided the probability of the feature value given the correct class by the probability of the feature value given the incorrect class, and took the log. The result was the contribution of the feature value for that particular case. We then averaged the attributions over all cases in each fold to achieve a single relevance score for the feature.

Maximum Entropy considers each feature with each value separately and therefore established a weight (relevance score) for each feature-value combination. Since we need a relevance score per feature rather than per feature-value combination, we calculated a weighted average relevance score for each feature, using the frequencies of the feature values as weights. The result was averaged over the 10 training sets. The model also shows which class is best selected for which feature value, enabling us to establish the preferred class when a feature is *present* (binary features) or relatively high (continuous features). Sometimes, the preferred class of a continuous feature varied per frequency range and no general trend could be detected.

The third metric is the feature ranking algorithm *Cluster Separation Score* (*CSS*), developed by Van Halteren. CSS is determined for each feature by dividing the difference between the means of the values with class *left* and class *right* by the sum of the standard deviations of the values with class *left* and class *right*. The resulting relevance score is an indication of the extent to which the feature is able to distinguish the cases with class *left* from those with class *right*. As with the model of Maximum Entropy, the formula shows which class is best selected for which feature value. *CSS* requires that the feature values are continuous, which was problematic for our data since the great majority consists of discrete (nominal) features. We converted these features (such as the presence or absence of a POS tag) to numerical features with values *0* and *1*. We assumed that despite the fact that the features are not truly continuous, the metric will still be able to estimate the relevance of the features.

Since it is undesirable to draw conclusions on features that occur in only one partition of the data, we removed those from the three rankings found. They are features that only have the values *absent* and *not applicable* (for binary features)

or *0* (for continuous features) in nine or ten partitions. From the total of 8,664 features[8], 806 features[9] remained after this removal.

In order to reach a final ranking of the 806 features, we averaged the rankings in the three methods. In these rankings, we have given features with equal ranking scores the same rank by averaging over their positions (e.g. rank 4.5 for two equally useful features at positions 4 and 5).

4.3.2 Results

Since an overview of all 806 features would be too extensive to suit this paper and would include a discussion of irrelevant features at the bottom of the list, we limit ourselves to the 50 best features following our ranking. Note that the features have either been determined for all three (M-)SDUs *x*, *y* or *z*, or for both (M-)SDU pairs *x*-*y* and *y*-*z* in the triple. Therefore, the features have forms such as *the (x)*, being the presence of the word *the* in *x*, or *anaphora (y-z)*, being the presence of an anaphoric relation between *y* and *z*. This section presents the findings for each feature type. The top 10 can be found in Table 4.3. The last column shows the range of the ranks found with the three different metrics.

Table 4.3: Top 10 of 806 features

	Feature	*Pos.*	*Feature type*	*Av. rank*	*Range*
1	pers. pronoun in first clause	*z*	reference	2	1–3
2	cont. quotation marks	*y-z*	discourse	3	1–6
3	word similarity (Lin)	*y-z*	lexical	4	4–4
4	word similarity (Lin)	*x-y*	lexical	7	5–11
5	token overlap	*y-z*	lexical	10	7–17
6	missing modifier	*y-z*	reference	12	2–23
7	def. article in first clause	*z*	reference	17	11–20
8	proper noun sg.	*z*	surface	17	9–22
9	proper noun sg.	*y*	surface	20	11–36
10	past tense	*x*	syntactic	21	12–36

Surface features

Of the 579 surface features, 10 features are in our top 50 (being words and POS tags). The trigrams, which were also included as surface features, are not present in the top 50. This is probably because of the small size of our data set.
The following word features are included in the top 50: *as (y)*, *farmer (z)*, *little (y)*, *the (z)* and *to (y)*. The presence of the word *farmer* in this top 50 is probably

[8] 7,828 different surface, 20 syntactic, 718 lexical, 84 reference and 14 discourse features.
[9] 579 surface, 20 syntactic, 136 lexical, 61 reference and 10 discourse features.

caused by the specific text type and data set[10]. The word *little* in *y* seems to refer back to *x*, because the relation for this feature is expected between *x* and *y* by the metrics of Maximum Entropy and CSS:

> *$_x$[If the pound falls closer to 2.80 marks, the Bank of England may raise Britain's base lending rate by one percentage point to 16%, says Mr. Rendell.]$_x$ – $_y$[But such an increase, he says, could be viewed by the market as "too little too late."]$_y$ $_z$[The Bank of England indicated its desire to leave its monetary policy unchanged Friday by declining to raise the official 15% discount-borrowing rate that it charges discount houses, analysts say.]$_z$*
> — wsj_0693

The relevance of the definite article *the* in *z* (ranked 15th) seems to confirm our intuition that *the* can be used as a reference word, and that references are important in discourse. However, in cases where *the* is present in *z*, CSS expects a relation between *x* and *y*, not between *y* and *z*, as in the example below:

> *$_x$[He made numerous trips to the U.S. in the early 1980s, but wasn't arrested until 1987 when he showed up as a guest of then-Vice President George Bush at a government function.]$_x$ – $_y$[A federal judge in Manhattan threw out the indictment, finding that the seven-year delay violated the defendant's constitutional right to a speedy trial.]$_y$ $_z$[The appeals court, however, said the judge didn't adequately consider whether the delay would actually hurt the chances of a fair trial.]$_z$*
> — wsj_0617

Apparently, *the* is more often used to introduce a new topic or argument in the text than to refer back to the previous (M)SDU. Journalists of the Wall Street Journal probably assume that readers are familiar with certain notions and topics (in this case *the appeals court*), thus mentioning them with the definite article.

POS tags that are in the top 50 are: *personal pronoun (z)*, *proper noun singular (y,z)*[11] and *third person singular verb (x)*. Personal pronouns are cues that the (M-)SDU in which it appears is rhetorically related to the previous (M-)SDU. Proper nouns are common in financial newspaper texts. The metrics of Maximum Entropy and CSS show that when a proper noun is present in the *z*, the relation is most likely between *x* and *y*, and when in *y*, between *y* and *z*. Apparently, a person or company often introduces a new topic. According to CSS, a *third person singular verb* in *x* is an indicator of a relation between *y* and *z*, while no consistent pattern is displayed by the metric of Maximum Entropy. Its presence in the top 50 may hint at the relevance of syntactic structure, as we will also find below.

The last surface feature present in the top 50 is the length (in words) of *y*. CSS and Maximum Entropy disagree on the direction of the rhetorical relation for this feature.

---

[10]The word *farmer* is quite frequent in the RST Treebank (and consequently in our data): in triple-final position (*z*), it is present in 25 triples extracted from 9 different paragraphs in 3 different texts in the Treebank.

[11]The notation *proper noun singular (y, z)* represents two features: *proper noun singular (y)* and *proper noun singular (z)*.

Syntactic features

The top 50 includes 5 of the 20 syntactic features. Both *present (x)* and *past (x)* tense are present in the top 50. Also included are *gerunds (y)* and *infinitives (y)*. The relatively high ranking of these syntactic features seems to indicate that syntactic structure is related to discourse structure.

Syntactic similarity is also in the top 50, but only for the left pair (*x*-*y*). CSS expects a rhetorical relation on the left when the syntactic similarity between *x* and *y* is high. This is in accordance with the literature and our intuitions based on inspections of the data. For Maximum Entropy, the direction depends on the similarity range: the expected class varies per interval (in the discretized version of syntactic similarity), and no general trend can be found.

Lexical features

Of the 718 lexical features, 12 belong to the 50 best features. Despite the fact that cue phrases are used in all systems discussed in the beginning of this paper, none of LeThanh's (2004) cue phrases come forward in our approach. NP cues (also taken from LeThanh) present in the top 50 are *speculation (x)* and *goal (y)*, and VP cues *affect (y)*, *assume (y)*, *have to (y)* and *bring (z)*. An (M-)SDU with *goal*, *affect*, *assume* or *have to* is most likely rhetorically related to the following (M-)SDU. The presence of *speculation* in *x* indicates a relation between *y* and *z*. Apparently, it is used to close a topic. The verb *bring* appears to introduce a new topic, since a relation is expected between *x* and *y* when *bring* is present in *z*.

Word overlap is ranked in the top 50 only for the right pair in the triple (*y*-*z*). A relatively high word overlap implies there is a rhetorical relation between the two (M-)SDUs concerned, which is what we expected.
The same expected pattern is found for *word similarity Lin (x-y, y-z)*. The higher the similarity, the higher the chance that a rhetorical relation exists. Word similarity on basis of WordNet is not present in the top 50. It is commonly known that the wide coverage of WordNet may lead to problems when applied to specific domains such as financial newspaper texts. Because Lin's Thesaurus was trained on Wall Street Journal texts, it is not surprising that the similarity based on Lin's Thesaurus is more useful for our task than the similarity based on WordNet.

Time references are only useful enough to be in the top 50 when they occur in *z*. According to both CSS and the model of Maximum Entropy, the presence of a time reference in *z* indicates that the relation is probably between *x* and *y*. This would mean that time references introduce new topics that are not rhetorically related to the previous (M-)SDUs, for example as in:

> *$_x$[Witnesses have said the grand jury has asked numerous questions about Jacob F. "Jake" Horton, the senior vice president of Gulf Power who died in the plane crash in April.]$_x$ —*
> *$_y$[Mr. Horton oversaw Gulf Power's governmental-affairs efforts.]$_y$*
> *$_z$[On the morning of the crash, he had been put on notice that an audit committee was recommending his dismissal because of invoicing irregularities in a company audit.]$_z$*

— wsj_0619

Note, however, that this example differs from the example in Section 4.2.1 where both *(M-)*SDUs in the relation contain a time reference. In order to also capture such instances, the presence of time references in two adjacent (M-)SDUs is best included as a separate feature in future research.

Reference features

Reference features are the most frequent (18) in the top 50[12]. The best feature according to our ranking method is a reference feature: a *personal pronoun in the first clause (y-z)*. The same feature but in *x-y* can be found at place 36 in the ranking. Also in the top 10 is the feature *definite article in the first clause (y-z)*. As we already saw in the discussion of the surface feature *the* above, the presence of a definite article in the first clause of *z* indicates that the relation is expected between *x* and *y*. Apparently, definite articles are most often used to refer to what is assumed to be known, not to what has previously been mentioned in the article.

Still, most reference features in the top 50 confirm our intuitions that anaphoric references in different forms are cues for rhetorical relations. When there is an *anaphoric relation (x-y, y-z)* between two (M-)SDUs, it is an indication that there is a rhetorical relation. A high *relative number of demonstrative (y-z)* and *personal pronouns (x-y, y-z)*, and the *presence of personal pronouns (x-y, y-z)* also predict a rhetorical relation with the preceding (M-)SDU. The relevance of personal pronouns has already been found in the surface features, as discussed above. Again, the definite article is especially used to introduce a new topic: a relatively high *number of definite articles* in *z* is a predictor of a relation between *x* and *y*.

Of the reference words in the top 50 (*added (x-y, y-z)*, *further (x-y)*, *less (x-y, y-z)* and *other (x-y, y-z)*), only *further* is not used to refer to the previous (M-)SDU. In our data, *further* seems to ask for an elaboration, for example:

> $_x$*[Operating profit grew 57% to* $269 million from 171$ *million, while operating margins rose to 16.1% from 15.9% the previous quarter and 12.6% a year ago.]*$_x$ $_y$*[Daniel Akerson, MCI chief financial officer, said the company sees further improvements in operating margins.]*$_y$ $-$ $_z$*["We think we can take it to the 18% range over next 18 to 24 months," he said.]*$_z$
> — wsj_1999

The last reference feature we defined, NP simplification, is present in the top 50 in the form of *missing modifiers (y-z)* and *missing head (x-y)*. When there are missing modifiers or the head is missing, a relation is indeed expected in that (M-)SDU pair.

[12] In our feature set, reference features are always determined for (M-)SDU *pairs*. Since we expect reference items to refer back, features such as the presence of reference words or of a personal pronoun in the first clause always concern the second (M-)SDU of a pair.

Discourse features

The top 50 includes 4 of the 14 discourse features available. The feature describing the position of the (M-)SDU in the paragraph (*sentence number in the paragraph (y, z)*) is included. We believe the relevance of this feature can be explained by the general structure of paragraphs in newspaper articles. This newspaper structure also makes itself felt in the feature *internal discourse structure (z)*. Testing these features on different text genres is necessary to establish whether our intuitions about newspaper structure are valid.

As expected, the presence of *continuous quotation marks (y-z)* is a cue for the presence of a rhetorical relation in both CSS and the model of Maximum Entropy.

4.3.3 Discussion

In the ranking, we have seen that some features are highly ranked by two of the three algorithms, but lower by the third. This explains the sometimes wide ranges in the last column of Table 4.3. Note that the average rankings are still relatively good given the large number of features (806). The best feature, *personal pronoun in first clause*, appears to be important for all three algorithms.

The list of best 50 features following our ranking strategy contains 'positive' features (expecting a rhetorical relation) as well as 'negative' features (introducing a new item in the text). Positive features are syntactic similarity, word overlap, word similarity (following Lin's (1998) Dependency Thesaurus), continuous punctuation and almost all reference features. Negative features include time references, proper nouns, definite articles, the word *further* and the verb *bring*. Obviously, both positive and negative features are useful for discourse analysis.

Some features unexpectedly come forward from our approach as relevant. The word *farmer*, for example, is likely to be dependent on the data set and therefore a bad predictor. Similarly, the high ranking of discourse features such as the position of (M-)SDUs in the paragraph and their internal discourse structure is probably caused by the general (financial) newspaper structure of the data. Results such as these can be prevented in future by extending the data with more texts of different genres. This may be difficult since no such data are available with discourse (i.e. RST) annotations yet.

4.4 Conclusion

In this paper, we have aimed at answering the question "Can we identify features that can be used to predict the presence of rhetorical (RST) relations between (Multi-)Sentential Discourse Units within paragraphs in English?" By reducing RST parsing to a classification problem, using an inventory of potentially relevant features (Section 4.2) and ranking them on the basis of three different metrics (Section 4.3), we have succeeded in this.[13] Some of the relevant features we have

[13]The feature values, the Perl scripts and the feature relevance scores found can be downloaded from `http://lands.let.ru.nl/~daphne`.

found predict the presence of a rhetorical relation (e.g. word similarity), while others are more often used to introduce new topics or arguments (the definite article for example).

In our research, we have reduced discourse analysis to a rather artificial classification task that is only a first step towards automatic discourse analysis. Also, we have limited ourselves to existing implementations of algorithms and metrics, without adjusting the parameters and without closely examining their capability to deal with our data. As this may have resulted in low accuracy and therefore only speculative rankings, we advise other researchers who plan to use our ranking to test the features on the real task with systems and settings that are more tuned to this kind of data.

References

Baayen, R.H., R. Piepenbrock, and L. Gulikers (1995), The CELEX Lexical Database (CD-ROM), *Technical report*, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA, USA.

Bernstein, Y. and J. Zobel (2005), Redundant documents and search effectiveness, *Proceedings of the 14th ACM international conference on Information and knowledge management*, ACM Press, New York, Bremen, Germany, pp. 736–743.

Black, E., S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski (1991), Procedure for quantitatively comparing the syntactic coverage of English grammars, *Proceedings of the workshop on Speech and Natural Language*, Leiden, the Netherlands, pp. 306–311.

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

Cohen, J. (1960), A coefficient of agreement for nominal scales, *Educational and Psychological Measurement* **20** (1), pp. 37–46.

Corston-Oliver, S.H. (1998), *Computing Representation of Discourse Structure*, PhD thesis, Dept. of Linguistics, University of California, Santa Barbara, CA, USA.

Demsar, J., B. Zupan, and G. Leban (2004), Orange: From Experimental Machine Learning to Interactive Data Mining, *Technical report*, Faculty of Computer and Information Science, University of Ljubljana. Software available at `http://www.ailab.si/orange`.

Kononenko, I. (1994), Estimating Attributes: Analysis and Extensions of RELIEF, *Proceedings of the European Conference on Machine Learning*, pp. 171–182.

LeThanh, H. (2004), *Investigation into an Approach to Automatic Text Summarisation*, PhD thesis, Middlesex University, UK.

Lin, D. (1998), Automatic retrieval and clustering of similar words, *Proceedings of the 17th international conference on Computational linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 768–774.

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

Marcu, D. (1999), A decision-based approach to rhetorical parsing, *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, College Park, Maryland, USA, pp. 365–372.

Marcu, D. (2000), The rhetorical parsing of unrestricted texts: a surface-based approach, *Computational Linguistics* **26** (3), pp. 395–448.

Marcus, P.M., M.A. Marcinkiewicz, and B. Santorini (1993), Building a large annotated corpus of English: the Penn Treebank, *Computational Linguistics* **19** (2), pp. 313–330.

Pedersen, T., S. Patwardhan, and J. Michelizzi (2004), WordNet::Similarity – Measuring the Relatedness of Concepts, *Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, MA, USA, pp. 38–41.

Poesio, M. and M. Alexandrov-Kabadjov (2004), A general-purpose, off the shelf anaphoric resolver, *Proceedings of 4th International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal.

Soricut, R. and D. Marcu (2003), Sentence Level Discourse Parsing using Syntactic and Lexical Information, *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Edmonton, Canada.

Theijssen, D. (2007), *Features for automatic discourse analysis of paragraphs*, Master's thesis, Radboud University Nijmegen, The Netherlands. Available at `http://lands.let.ru.nl/~daphne/MA_thesis.html`.

Zhang, L. (2004), *Maximum Entropy Modeling Toolkit for Python and C++*.

5

Detecting semantic overlap

A parallel monolingual treebank for Dutch

Erwin Marsi and Emiel Krahmer
Dept. of Communication and Information, Tilburg University

Abstract

This paper describes an ongoing effort to build a large-scale monolingual treebank of parallel/comparable Dutch text, where nodes of syntax trees are aligned and labeled according to a small set of semantic similarity relations. Such a corpus has many potential uses and applications in e.g., multi-document summarization, question-answering and paraphrase extraction. We describe the text material, preprocessing, annotation, and alignment of sentences and syntax trees, both manual and automatic. Two new annotation tools are presented, as well as results from pilot experiments on inter-annotator agreement. On the basis of this resource, new automatic alignment software and NLP applications will be developed.

5.1 Introduction

We describe an ongoing effort within the context of the DAESO (Detecting And Exploiting Semantic Overlap) project[1] to build a large-scale monolingual treebank of parallel/comparable Dutch text consisting of over 1 million words, where nodes of syntax trees are aligned and labeled according to a small set of semantic similarity relations. This paper aims to attract and inform potential users of this resource, providing information about its design, potential applications, development process, current status, and plans for deriving software tools and NLP applications.

[1] For the latest developments, see the website at http://daeso.uvt.nl

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 69–84
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: e.c.marsi@uvt.nl

In the first part of the paper we introduce the two key ideas of a parallel monolingual treebank and semantic similarity relations. We discuss potential applications in multi-document summarization, question-answering, information extraction and textual entailment. The remainder of the paper focuses on the construction of the corpus. We describe the raw text material, preprocessing, tokenization and syntactic parsing. Next is alignment at the sentence level, both automatic and manual, and finally alignment of syntax trees. We introduce newly developed annotation tools and present results from pilot experiments on inter-annotator agreement. We finish with a summary and plans for future work on automatic alignment software and NLP applications.

5.2 Background

5.2.1 Parallel monolingual treebanks

Treebanks of syntactically annotated sentences have become an essential resource in computational linguistics and related areas. Not only for developing and systematically validating computational models of syntax, but also for data-driven development of natural language processing tools such as part-of-speech taggers, chunkers and parsers. In a similar vein, large *parallel corpora* of *bilingual* text have become the basis for statistical and example-based machine translation. Typically, the text material in a bilingual parallel corpus is aligned at the level of sentences, words or arbitrary substrings. Several researchers have begun to explore *parallel treebanks* with more linguistically motivated information in the form of aligned phrase-structure trees or dependency structures (see e.g., Gildea (2003) and Samuelsson and Volk (2006)).

A similar type of resource, parallel corpora of *monolingual* text, have proved useful for automatic extraction of synonyms and paraphrases, which in turn has a wide range of applications from machine translation to information retrieval. This has also inspired work on *comparable corpora* of loosely associated text like comparable entries from different encyclopedias (Barzilay and Elhadad 2003).

A logical combination of these two trends – parallel bilingual treebanks on the one hand and monolingual parallel/comparable text corpora on the other – leads to the idea of a *parallel monolingual treebank*, which we define as a corpus of parallel/comparable text in the same language with aligned parse trees. It seems that so far little or no published research has addressed this notion – but see (Ibrahim et al. 2003). In our opinion parallel monolingual treebanks hold great potential, not only for paraphrasing, but also in general for studying the mapping from meaning to alternative surface realizations, and in many NLP applications.

5.2.2 Tree alignment and semantic similarity relations

Alignment of syntax trees is the process of aligning those pairs of nodes that are similar. More precisely: for each node v in the syntactic structure of a sentence S, its yield $\textsc{str}(v)$ is defined as the substring of all tokens under v (i.e., the composition of the tokens of all nodes reachable from v). An alignment between sentences

S and S' then pairs nodes from the syntax trees for both sentences. Aligning node v from the syntax tree T of sentence S with node v' from the tree T' of sentence S' means that there is a similarity relation between their yields $\text{STR}(v)$ and $\text{STR}(v')$.

This makes node alignments to some extent similar to the dependencies among words in a syntactic dependency structure. However, where dependencies are normally labeled in terms of a set of meaningful dependency relations, alignments are unlabeled in the majority of work on alignment. If labeled, the set of relations is limited to a binary distinction like *sure* and *possible* (Daumé III and Marcu 2005), or *good* and *fuzzy* (Volk et al. 2006).

In contrast, we propose to label alignments according to a small set of *semantic similarity relations*. By way of example, we use the following pair of Dutch sentences:

1. Dagelijks koffie vermindert risico op Alzheimer en Dementie.
   Daily coffee diminishes risk on Alzheimer and Dementia.

2. Drie koppen koffie per dag reduceert kans op Parkinson en Dementie.
   Three cups coffee a day reduces chance on Parkinson and Dementia

The corresponding syntax trees and their (partial) alignment is shown in Figure 5.1. We distinguish the following five mutually exclusive similarity relations:

1. v **equals** v' iff $\text{STR}(v)$ and $\text{STR}(v')$ are literally identical (abstracting from case). Example: "Dementie" equals "Dementie";
2. v **restates** v' iff $\text{STR}(v)$ is a paraphrase of $\text{STR}(v')$ (same information content but different wording). Example: "risico" restates "kans";
3. v **generalizes** v' iff $\text{STR}(v)$ is more general than $\text{STR}(v')$. Example: "dagelijks koffie" generalizes "drie koppen koffie per dag";
4. v **specifies** v' iff $\text{STR}(v)$ is more specific than $\text{STR}(v')$. Example: "drie koppen koffie per dag" specifies "dagelijks koffie";
5. v **intersects** v' iff $\text{STR}(v)$ and $\text{STR}(v')$ share some informational content, but also each express some piece of information not expressed in the other. Example: "Alzheimer en Dementie" intersects "Parkinson en Dementie"

It should be noted that for expository reasons the alignment shown in Figure 5.1 is not exhaustive.

5.2.3 Applications

Aligning syntax trees and labeling alignments in terms of these semantic similarity relations has many interesting theoretical and practical implications, some of which are explained below.

Sentence fusion in multi-document summarization Given a set of similar documents, a multi-document summarization system must first identify the most important sentences for inclusion in the summary. To avoid redundancy, the system

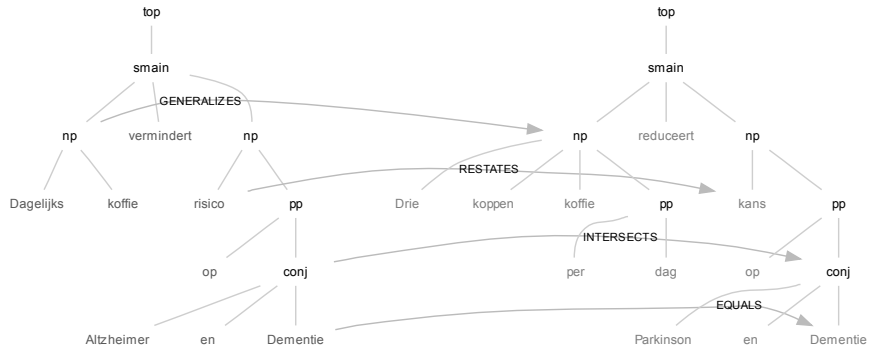

Figure 5.1: **E**xample of two (partially) aligned syntactic trees

must be able to detect similar sentences, which amounts to the task of sentence alignment in comparable texts. Summarizers which attempt to produce real summaries – instead of merely extracts – must also revise sentences. On the one hand, they must identify and merge similar information in order to avoid redundancy. On the other hand, they must remove unimportant parts as well as rephrase parts in order to accomplish further sentence compression. One can envision this revision process as one of aligning, merging and pruning syntax trees, followed by the generation of revised sentences using techniques from Natural Language Generation – an approach called *sentence fusion* by Barzilay and McKeown (2005). Our labeled alignments are an interesting extension here, because they allow a system to generate fused sentences which are more specific, equivalent or more general than the original ones. Some of our initial work in this area is described in (Marsi and Krahmer 2005a).

Clustering answers in Question-Answering Question-Answering (QA) systems typically analyze a question, search for potential answers in a large body of text material, produce a list of potential answers ranked in order of decreasing likelihood, and show only the topmost answer to the user. For questions of the "open" type, like "What are the risks of overweight?", the topmost answer is unlikely to be optimal. On the one hand, it may be incomplete in the sense that it does not exhaustively list all the risks of overweight encountered in the full text collection. On the other hand, as it is a piece of text extracted from a particular context, it may contain additional information which is irrelevant to the question. We think that detecting and merging similar answers will lead to answers which are both more comprehensive and more to the point. An experiment addressing users' preferences for several different types of answer fusions is described in (Krahmer et al. 2008).

Paraphrases in Information Extraction A large corpus of aligned monolingual text can be used to extract paraphrases from. Here we are interested not so much in synonyms, many of which are already available from other resources like Wordnet, but rather in complex paraphrases as in the pair "X left company Y" and "X, former employee of Y". These may be extracted from aligned syntax trees along the lines of (Lin and Pantel 2001). We plan to investigate to what extent structural paraphrases of this type improve tasks in information retrieval and information extraction.

Recognizing Textual Entailment Recognizing Textual Entailment (RTE) is the task of predicting whether a certain sentence (the hypothesis) is entailed by a one or more other sentences (the text). It is proposed as a general NLP task, akin to Word Sense Disambiguation or Named Entity Recognition, as a common building block for applications like summarization, QA, IR and IE (Dagan et al. 2005). Many RTE systems use a form of alignment under the assumption that entailment is likely if the hypothesis can be aligned with the text. An example of this approach using alignment of syntax trees is described in (Marsi et al. 2006).

5.3 Corpus material and annotation

5.3.1 Text material

The corpus contains Dutch text material from five different sources. The target size of the final corpus is 1 million words, half of which is processed with partly manual annotation and correction, whereas the other half will be processed fully automatically. The scope of the current discussion is limited to the first half million.

The composition of the corpus is the result of a trade-off between several constraints. To begin with, we intended to cover the range from true *parallel* text down to loosely associated *comparable text*, preferably across different text genres too. Furthermore, several text types are motivated by potential applications in summarization and QA. Finally, we needed to be able to settle copyright issues in a proper legal manner to ensure that the corpus can be made available to other researchers. This resulted in the following five text components.

Book translations True parallel text was obtained from alternative Dutch translations of foreign language books (about 125k words). Choices for source material were limited to older books, because alternative translations of recent books are extremely hard to come by. The DAESO corpus includes parallel Dutch translations from (parts of) three books: (1) "Le Petit Prince" by Antoine de Saint-Exupéry, (2) "On the Origin of Species" by Charles Darwin (in the 1st and 6th edition) and (3) "Les Essais" by Michel de Montaigne. Admittedly, the latter two books contain dated language use, but we used recent translations in modern Dutch.

Autocue-subtitle pairs This material comes from the *NOS journaal*, the daily news broadcast by the Dutch public television. It consists of the autocue text as read by the news reader and the associated subtitles (125k words). It was tokenized and aligned at the sentence level in the ATRANOS project (Daelemans et al. 2004). Because of space constraints, the subtitles typically present a compressed form of the autocue, making it an excellent source for work on automatic text compression, one of the subtasks in summarization.

News headlines Another source consists of similar headlines from online news articles. These were automatically mined from the Dutch version of Google News (25k words). As the site's clustering is based on the full article content rather than only the headline, we often found substantial differences between headlines. We therefore performed a manual subclustering in order to eliminate outliers and to extract sets of sufficiently similar sentences.

QA-system output For future work aimed at question-answering, the corpus also contains samples from the QA domain. The IMIX project has developed a multimodal QA system in the medical domain (Theune et al. 2007), where questions are answered by searching a large collection of text ranging from medical encyclopedias to layman websites. In order to evaluate the QA engines, a reference corpus of questions and associated answers as encountered in the available texts was manually compiled. From this corpus, we extracted all clusters of two or more alternative answers. With about 1k words, this segment is relatively small.

Press releases As the main source for comparable text, we used press releases regarding the same news event (225k words). These were obtained from news feeds of ANP and Novum, two Dutch press agencies. This type of material is particularly suitable to bootstrap automatic multi-document summarization. Similar articles were automatically extracted within a certain time window, relying on simple word overlap measures. The automatic procedure aimed at a high recall at the expense of precision, but was followed by manual correction. The reason is that finding similar articles in two large collections is much harder, for humans at least, than deciding whether a given pair of articles is indeed similar.

5.3.2 Preprocessing

Preprocessing in general involved converting all text material to XML format with UTF-8 character encoding. All book translations were (mostly) automatically converted from their original electronic format (raw text, MS Word, PDF) to XML adhering to the *TEI Lite* standard, the light version of the Text Encoding Initiative markup language (Burnard and Sperberg-McQueen 2006). The original document structure and formatting was preserved as much as possible in the markup; at a minimum, all books have markup indicating chapters and sections. In addition, manual markup was added to indicate parts of the texts which were not fit for our purposes, e.g., citations in a foreign language.

As TEI Lite is not particularly suited for the markup of the other types of text material, these were converted to custom XML formats. For instance, the XML for the headlines stores information regarding timestamps and sources, and indicates clusters and subclusters.

5.3.3 Tokenization

All text material was subsequently tokenized using the Dutch tokenizer developed within the D-COI project (Martin 2007), with the exception of the autocue-subtitle material, which was already tokenized. Sentence boundaries were indicated by inserting additional sentence tags in the XML source files, where each sentence element was assigned a unique id attribute.

We found that tokenization errors were more frequent in the book material, presumably because of long and complex sentences. As in particular sentence splitting errors will be fatal for the subsequent parsing step, and because tokenization errors are relatively cheap to fix, we undertook manually correction in this corpus segment. Tokenization errors in the press releases where fixed only in so far as they were noticed by the annotators during the subsequent step of sentence alignment.

5.3.4 Syntactic parsing

Next, the Alpino parser for Dutch (Bouma et al. 2001) was used to parse suitable sentences, excluding e.g., chapter heading, footnotes, citations in a foreign language, etc. The parser provides a relatively theory-neutral syntactic analysis as originally developed for the Spoken Dutch Corpus (van der Wouden et al. 2002). It is a blend of phrase structure analysis and dependency analysis, with a backbone of phrasal constituents and arcs labeled with syntactic function/dependency labels. Output is stored as a *treebank*, which is simply a list of parses in the Alpino XML format, with an id attribute identical to that of the input sentence in the source document.

As no parser is perfect, parsing errors, in the form of partly erroneous or fragmented analyses, are unavoidable. In addition, a few very long and/or particularly ambiguous sentences failed to pass at all. Due to time and cost constraints, such parsing errors were not subject to manual correction.

5.4 Sentence alignment

The next stage involved aligning similar sentences. Part of the text material was already aligned at the sentence level: the autocue-subtitle segment was aligned within the ATRANOS project; the alternative answers from the QA reference corpus are implicitly aligned, and the same goes for all sentences in subclusters of news headlines. Hence alignment of sentences was required only for the book translations and the press releases. This process was carried out in two steps: automatic alignment followed by manual correction.

5.4.1 Automatic alignment of parallel translations

Automatic alignment of sentences from parallel translations is a well-studied area for which a number of standard solutions are available, e.g., (Gale and Church 1993). It is usually assumed that the bulk of the alignments is of the 1-to-1 type, and that crossing alignments and unaligned sentences are rare. We found that these assumptions are frequently violated in our samples. For instance, in the two translations of "On the Origin of Species" from different editions, there are many differences, largely due to Darwin's own revisions. These range from long sentences in one translation being split in multiple sentences in the other to substantial pieces of added or removed text (the 6th edition even has a whole new chapter).

As the initial automatic alignment was known to be manually corrected afterward, we opted for a straightforward pragmatic approach that is robust to above problems. It takes a sentence from the first translation and checks for all sentences in a sliding window over the second translation at approximately the same position whether two sentences are sufficiently similar to justify alignment, where similarity is defined in terms of n-gram overlap. We use a relatively low threshold to get a high recall at the expense of precision, because in practice manually deleting unintended alignments is an easier task than identifying all correct ones.

Obviously, this approach is sensitive to large gaps due to insertion/deletion of substantial pieces of text. We therefore found it beneficial to carry out automatic alignment in multiple passes. That is, first align chapters, next align sections, then paragraphs and finally sentences.

Alignments were stored in a simple custom XML format which contains two types of information: (1) references to the source and target documents; (2) a list of links specifying the tags and id's of the aligned source and target elements respectively. We will refer to an XML document containing this information as a *parallel text corpus*.

5.4.2 Manual correction of sentence alignments

A special alignment annotation tool, called *Hitaext*, was developed for visualizing and editing alignments between text segments.[2] It takes as input a parallel text corpus, i.e. an XML file defining source and target XML documents plus a list of links between elements in those two documents. It then offers two different views on the input documents: the tree view and the text view.

The *tree window* – see the left side of Figure 5.2 – visualizes the hierarchical structure of the XML elements in the form of a pair of adjacent tree controls (or tree widgets). These allow a user to quickly walk through the document structure using mouse and/or keyboard. In our case, these documents may be TEI Lite XML documents, where the elements correspond to chapters, section, paragraphs and sentences. One of the advantages of this representation is that large documents

[2]Hitaext is implemented in wxPython, runs on Mac OS X, Linux and Windows, and is released as open source software from http://daeso.uvt.nl/hitaext

Figure 5.2: Screen shot of Hitaext, the tool used for aligning text segments

remain manageable because the user can expand or collapse arbitrary parts of the document tree. In addition, irrelevant elements can be either skipped or hidden completely by configuring the style sheet.

The *text window* – see the right side of Figure 5.2 – shows the two pieces of text corresponding to the two elements currently in focus (or selected) in the tree window. These are typically the texts of sentences, paragraphs, chapters or even whole documents (when the focus in on the root node). The sliders at the bottom of the text window allow the user to reveal a variable amount of the context.

If an element is aligned, its tag is shown in green in the tree window and its corresponding text is also shown in green in the text window. In contrast, unaligned tags and texts are shown in red. If a focused element has alignments, the aligned elements in the other tree are marked by means of an exclamation mark icon. This representation accommodates 1-to-1, 1-to-n and n-to-m alignments. By selecting two elements from either tree and subsequently hitting the space bar, a user can toggle (switch on/off) the alignment between them.

One of the distinguishing features of Hitaext in comparison with other text alignment tools is its support for simultaneous alignment at multiple annotation levels (e.g., words, sentences, paragraphs, and chapters). Another feature is the lack of a predefined input format for the source and target documents; the only constraint is that the XML must be well-formed. In fact, this makes Hitaext a general graphical tool for aligning text elements in pairs of arbitrary XML documents. It might therefore as well be used for tasks like aligning bilingual text at the word level or aligning ontological hierarchies. Other features include automatic synchronization of the two trees, i.e. focus automatically jumping to the (first) aligned element in the other tree, and quickly walking through all aligned elements in the case of one-to-many or many-to-many alignment.

5.4.3 Alignment of comparable text

The assumptions about parallel text clearly do not hold in the case of comparable text: one-to-many alignments – or even many-to-many – are to be expected frequently, just as crossing alignments and large portions of unaligned material. Moreover, similarity between sentences, and therefore the decision to align or not, is much more gradient. Whereas with parallel translations it is virtually always evident whether or not two sentences are translations of the same source sentence, it is much harder to decide whether two comparable sentences are sufficiently similar to justify alignment. In order to preserve a reasonable level of consistency among annotators, we have developed a set of annotation guidelines.

For a start, there is no need for aligned sentences to be true paraphrases of each other. One sentence may contain pieces of information which are not present in the other. Likewise, information in one sentence may be more specific/general than in the other. However, aligned sentences should have at least one proposition in common, which we interpret informally as a statement about someone or something. Examples of (partial) sentences including the same proposition:

- Balkenende is the minister-president of the Netherlands
- Balkenende, who is the minister-president of the Netherlands, ...
- Balkenende, the minister-president of the Netherlands ...
- Balkenende as the minister-president of the Netherlands ...
- Balkenende being minister-president of the Netherlands ...

However, the following examples do not share a proposition (even though they may share some words):

- Balkenende is a wine expert
- Balkenende likes to barbecue
- Bush likes to barbecue
- the minister-president of the Netherlands
- the capital of the Netherlands

Furthermore, we do not attempt to align each sentence with *the* most similar sentence (one-to-one alignment). Instead, we align each sentence to every other sentence with which it has sufficient overlap, i.e. at least one proposition in common, effectively allowing many-to-many alignment.

We also allow use of common sense. Consider the following pair:

- Keith Urban left US rehabilitation clinic
- Keith Urban cured from addiction

In the strict logical sense, these statements differ: in theory, one may leave the clinic without being cured, or one may be cured but remain in the clinic. However, in the context of two texts on the same topic, we prefer to view them as similar for all practical purposes. This is in the same spirit as *natural entailment* is defined in the Recognizing Textual Entailment task (Dagan et al. 2005). Other examples

include approximately identical locations, quantities, times, etc. In a similar vein, referring expressions like pronouns or generic definite descriptions may be interpreted in the context.

However, we refrain from alignment in cases where inferring similarity requires elaborate reasoning and background knowledge, as in:

- The Radicals now hold 80 of the 250 seats in parliament
- The SRS is currently the biggest party in Serbia

Notice that this would requires one to know that 80 out of 250 is a majority because all other political parties hold fewer seats.

5.4.4 Inter-annotator agreement

We carried out a pilot experiment with two annotators who each aligned the same 10 pairs of comparable press releases, varying in length from 4 tot 33 sentences. The total number of possible one-to-one sentence alignments to consider was 1492. Both annotators agreed on 44 but disagreed on 32 alignments. While discussing the differences, we encountered some difficult cases which gave rise to revision of the annotation guidelines. However, it turned out that the majority of the disagreements were caused because an annotator simply failed to notice a particular alignment.

After revision of the guidelines, we repeated the experiment with another set of 10 comparable press releases, this time with 1337 possible alignments to consider. One annotator made 15 unique alignments and the other 39, in other words, they disagreed on 54 cases, while both agreed on 51 alignments. Even though the ratio is slightly worse than before, this time by far the most disagreements were caused by overlooked alignments. This supports our impression that the task of identifying *all* pairs of similar sentences is harder than deciding on similarity of a particular pair of sentences. In terms of precision and recall, this means that precision on sentence alignment is substantially better than recall.

5.5 Syntax tree alignment

The final stage consists of aligning the syntax trees according to the semantic alignment relations described in Section 5.2.2. For creating and labeling alignments, we developed another special-purpose annotation tool called *Algraeph*.[3] The screen shot in Figure 5.3 shows Algraeph with the sentence pair from example (1). The input sentences are shown in the text boxes at the top. The corresponding syntactic trees are shown in the middle, with alignments indicated by curved lines. In the interface, normal alignments are rendered in green, whereas focused nodes and their alignments are yellow. The token sequences corresponding to these focused nodes are shown in the text boxes at the bottom, with the alignment relation,

[3] Algraeph is a rewritten and extended version of our earlier tool called Gadget. It is implemented in wxPython, runs on Mac OS X, Linux and Windows, and is available as open source software from http://daeso.uvt.nl/algraeph

which is ”generalizes” here, in between. This relation can be changed or removed by clicking any of the radio buttons.

One of the challenges for annotators is dealing with the large syntax trees of long sentences, which are hard to navigate. Algraeph therefore has a range of options to tune the rendering of the trees and alignments.

Input to Algraeph is a *parallel graph corpus*, a file in a simple custom XML format which contains three types of information. First, one or more references to the treebanks containing the syntactic trees of the aligned sentences. Second, a list of linked trees which are identified by the id's of the source and target trees and the id's of the treebanks they originate from. Third, for each pair of linked trees, a list of linked nodes in terms of the id's of the aligned source and target nodes as well as the alignment relation.

A parallel graph corpus is automatically derived from the combination of a source and target text document, a parallel text corpus, and the corresponding treebanks. In order to speed up the manual annotation, alignments of the type “equals” are established automatically, as these can be be reliably detected by automatic procedures. Our aim is to bootstrap the annotation process so other alignment relation can also be automatically predicted and the manual annotation process will ultimately involve only correction work.

5.5.1 Inter-annotator agreement

As is to be expected, the correct alignment is not always evident. For example, because of differences in interpreting the meaning of a particular word or phrase, or because of blurry edges between the categories of semantic relations. Other issues arose because of parsing errors which prevent one from making the desired alignment. We are currently in the process of writing an annotation manual with guidelines to resolve these issues in a consistent manner. In general we adhere to the same principles we discussed earlier in the context of alignment of comparable sentences. For instance, referring expressions may be interpreted in the context and the use of common sense is allowed.

In (Marsi and Krahmer 2005a) we reported on a pilot experiment which involved aligning syntax trees from the first five chapters of “Le Petit Prince”. Results indicated that humans can perform this task well, with ultimately an F-score of .98 on creating alignments and an F-score of .95 on assigning semantic similarity relations. We also presented results on automatic annotation, which achieved an F-score on alignment of .85 and an F-score of .80 on semantic relation classification, albeit with some strong assumptions regarding prior knowledge.

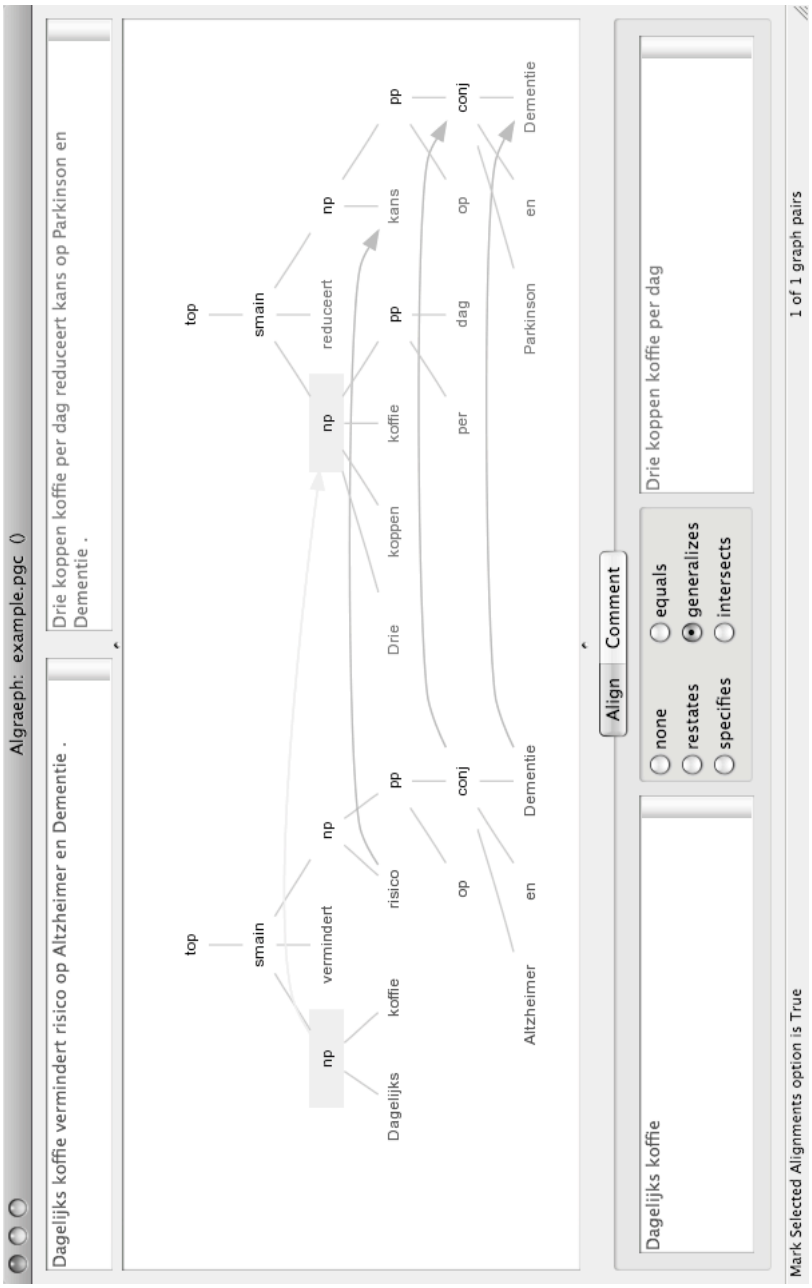

Figure 5.3: Screen shot of Algraeph, the tool used for aligning syntax trees

We repeated this experiment on a smaller scale with 30 sentence pairs from the same book translations, but this time with other annotators. Again the F-score on assigning semantic similarity relations was over .95. It should be noted though that due to the nature of the text material the majority of the alignments involved "equals", and therefore the agreement is relatively high. We expect lower figures when we repeat the experiment with other types of text material in the corpus. At any rate, the current corpus will allow us to repeat these experiments on larger scale and with more challenging text material in the near future.

5.6 Summary and future work

We have introduced the idea of parallel monolingual treebanks and of alignments labeled according to a set of five semantic similarity relations. We have suggested potential applications in multi-document summarization, QA, IE and RTE. Next, we described the ongoing effort to build a large-scale parallel/comparable monolingual treebank for Dutch of over 1 million words. We described the text material from different sources (book translation, autocue-subtitles, news headlines, answers from the QA domain, and press releases) and the process of preprocessing, tokenizing, and syntactic parsing. Next, we addressed alignment at the sentence level, both automatic and manual, followed by alignment at the syntax level. Two new graphical annotation tools were presented (Hitaext and Algraeph). Results from pilot experiments on inter-annotator agreements showed encouraging results. We are currently working on the manual alignment of syntax trees for the first half million words of the corpus.

Apart from the two graphical tools for manual alignment, we are developing software for automatic alignment of sentences from parallel and comparable text sources. Likewise, we will continue work on automatic alignment of syntax trees along the lines of (Marsi and Krahmer 2005b). Evidently, the corpus is an excellent resource for this. Once reasonably reliable tools for alignment are in place, we intend to double the size of our corpus to 1 million words by automatically aligning more text material, from the same sources in roughly equal proportions.

In addition, we intend to exploit our tools for detecting semantic overlap in a number of practical applications. In the context of multi-document summarization, we intend to take advantage of the semantic labeling of the alignments, which allows us to generate fused sentences which are more specific, equivalent or more general than the original ones. Some of our initial work in this area is described in (Marsi and Krahmer 2005a). The corpus segment containing QA answers will be used to bootstrap work on automatic clustering and fusing of similar answers from QA systems. Finally, we plan to measure the contribution of automatically derived structural paraphrases in IR and IE.

Acknowledgments We would like to thank Nienke Eckhardt, Paul van Pelt, Hanneke Schoormans and Jurry de Vos for all the annotation work, Erik Tsjong Kim Sang and colleagues for the autocue-subtitle material from the ATRANOS project, Gosse Bouma and others from the IMIX project for the QA reference corpus, and Wauter Bosma for mining the headlines from Google News. We also like to express our gratitude to the publishers

and press agencies for providing the raw text material. This work was conducted within the DAESO project funded by the Stevin program (De Nederlandse Taalunie).

References

Barzilay, R. and K.R. McKeown (2005), Sentence fusion for multidocument news summarization, *Computational Linguistics* **31** (3), pp. 297–328, Cambridge, MA, USA.

Barzilay, R. and N. Elhadad (2003), Sentence alignment for monolingual comparable corpora, *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pp. 25–32.

Bouma, G., G. van Noord, and R. Malouf (2001), Alpino: Wide-coverage computational analysis of Dutch, *Computational Linguistics in the Netherlands 2000*, Rodopi, Amsterdam, New York, pp. 45–59.

Burnard, L. and C. M. Sperberg-McQueen (2006), TEI Lite: Encoding for interchange: an introduction to the TEI Revised for TEI P5 release, *Technical report*, Text Encoding Initiative.

Daelemans, W., A. Höthker, and E. Tjong Kim Sang (2004), Automatic sentence simplification for subtitling in Dutch and English, *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pp. 1045–1048.

Dagan, I., O. Glickman, and B. Magnini (2005), The PASCAL Recognising Textual Entailment Challenge, *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, U.K.

Daumé III, H. and D. Marcu (2005), Induction of word and phrase alignments for automatic document summarization, *Computational Linguistics* **31** (4), pp. 505–530.

Gale, W.A. and K.W. Church (1993), A program for aligning sentences in bilingual corpora, *Computational Linguistics* **19** (1), pp. 75–102, MIT Press, Cambridge, MA, USA.

Gildea, D. (2003), Loosely tree-based alignment for machine translation, *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan, pp. 80–87.

Ibrahim, A., B. Katz, and J. Lin (2003), Extracting structural paraphrases from aligned monolingual corpora, *Proceedings of the second international workshop on Paraphrasing*, Vol. 16, ACL, Sapporo, Japan, pp. 57–64.

Krahmer, E., E. Marsi, and P. van Pelt (2008), Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion, *The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Columbus, Ohio, pp. 193–196.

Lin, D. and P. Pantel (2001), Discovery of inference rules for question answering, *Natural Language Engineering* **7** (4), pp. 343–360.

Marsi, E. and E. Krahmer (2005a), Explorations in sentence fusion, *Proceedings of*

the 10th European Workshop on Natural Language Generation, Aberdeen, GB, pp. 109–117.

Marsi, E. and E. Krahmer (2005b), Semantic classification by humans and machines, *ACL 2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan, pp. 1–6.

Marsi, E., E. Krahmer, W. Bosma, and M. Theune (2006), Normalized alignment of dependency trees for detecting textual entailment, *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy, pp. 56–61.

Martin, R. (2007), Sentence-splitting and tokenization in D-Coi, *Technical Report 07-07*, ILK Research Group.

Samuelsson, Y. and M. Volk (2006), Phrase alignment in parallel treebanks, *Proceedings of 5th Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republik.

Theune, M., B. van Schooten, R. op den Akker, W. Bosma, D. Hofs, A.Nijholt, E. Krahmer, C. van Hooijdonk, and E. Marsi (2007), Questions, pictures, answers: Introducing pictures in question-answering systems, *ACTAS-1 of X Symposio Internacional de Comunicacion Social*, Santiago de Cuba, pp. 450–463.

van der Wouden, T., H. Hoekstra, M. Moortgat, B. Renmans, and I. Schuurman (2002), Syntactic analysis in the Spoken Dutch Corpus, *Proceedings of the third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain, pp. 768–773.

Volk, M., S. Gustafson-Capkova, J. Lundborg, T. Marek, Y. Samuelsson, and F. Tidstrom (2006), XML-based phrase alignment in parallel treebanks, *Proceedings of EACL Workshop on Multidimensional Markup in Natural Language Processing*, Trento, Italy, pp. 93–96.

6

In response to your inquiry

Automatic e-mail answer suggestion in a Dutch Contact Centre

Michel Boedeltje and Arjan van Hessen
Telecats and Telecats/University of Twente

Abstract

In the past years, the number of service requests through e-mail has shown an explosive growth, forcing companies and government to set-up contact centres in order to handle these e-mails. Equal to most telephony services handled by call centres, 80% of the incoming e-mails is about 20% of the subjects, making it worthwhile to compose standard answers for at least the 20% most popular questions. Personal answering (each e-mail answered by a human agent) is simply too expensive and not necessary due to this 80-20 rule: well formed predefined answers can cover a significant part of the questions. By using IR and text classification techniques combined with Natural Language Processing, the process of finding the correct answer for a request can be (partly) automated. In this paper we will describe an answer suggestion system using IR based classification and NLP techniques. A practical study using an e-mail corpus of 17,000 incoming e-mails (collected and categorized in a Dutch contact centre), has shown that this approach is able to present the correct answer within a ranked list of 5 possible suggestions, for almost 88% of all incoming e-mails. Furthermore, we will show that this approach can be used as well for spoken content by combining the categorization techniques with the recognition result of the answer on the famous first question: ”Hello, how can we help you?”.

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 85–100
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: michel@telecats.nl

6.1 Introduction

With the ongoing acceptance of e-mail as a fast, cheap and reliable way of communication, companies receive an increasing number of service requests via e-mail. To handle these e-mails, call centres are ”transformed” into so-called contact centres handling both e-mail and telephone calls. Since most service requests cover a relatively small set of problems or questions (the Pareto effect, saying that 80% of the questions is about 20% of the subjects (Reed 2001)), many of these requests may be answered using a relatively small set of standard answers. Handling great amounts of e-mail in a contact centre is a very labour-intensive task, requiring a serious investment of time and money. Automating the answering process could therefore account for serious cost reduction and a significant decrease in response time. Due to the difficulty of automatically selecting the correct answer and thus the risk of sending the incorrect answer to a customer, most companies are reluctant to incorporate such an automatic e-mail answering system in their contact centres. However, automatically suggesting one or more relevant answers to incoming messages provides a good alternative. If we manage to suggest the correct answer in for instance a top-x of relevant answers, the agent only needs to browse through the selected answers and pick the correct one out of these x selected answers, instead of formulating the answer manually or searching the correct answer from all possible answers. Such a system would improve the efficiency in a contact centre and reduce the time spent on answering e-mail. However, one has to tune the number of suggested answers. Suggesting a small number of answers decreases the time an agent has to spend on browsing through the suggestions if the right answer is in these suggestions. However, if the right answer is not present in the suggestions, an agent has to spend extra time on manually searching the correct suggestion. Increasing the number of suggestions increases the chance the right answer is in the suggestions, but also increases the time an agent has to browse all these suggestions. Ideally, no more than 5 suggestions are given to the agents.

6.1.1 Problem statement

The contact centre where this research was performed uses an e-mail management system that enables contact centre agents to handle incoming e-mail efficiently. This system also provides functionality to automatically suggest relevant answers. This answer suggestion routine maps incoming e-mail to a standard question, based on the presence of predefined keywords in the incoming message. Each set of keywords is manually assigned to a standard question and the standard question that has the most keywords in common with the incoming e-mail, links to the best answer suggestion. Although this procedure works fine with a very homogeneous set of e-mails where each suggestion is defined by a well defined set of keywords, it turned out that in real world applications the variety of text in the e-mails is too high, to handle it well with keywords. In the studied case, there was a probability of just 50% that the right answer was present in the 10 best suggestions. So the agent had to browse through 10 suggestions with a chance of only

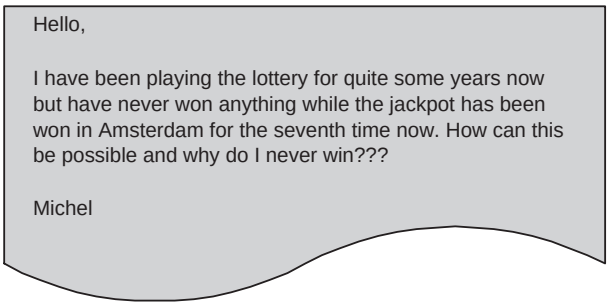

Figure 6.1: Typical e-mail in our contact centre

50% that the correct answer was in the suggestions. As a result, the system became useless. The main goal of this (practical) study was to investigate to what extent Information Retrieval based classification techniques improve the automatic suggestion of answers. In figure 6.1 we printed a typical e-mail (translated to English) for our contact centre of a well-known Dutch lottery.

6.1.2 Classification and speech recognition

Besides suggesting possible answers for written questions (e-mails), we will look at the possibility of using the discussed techniques in speech enabled call routing as well. Instead of confronting a calling customer with a confusing and elaborate IVR[1] menu, we prompt the caller to just say why they call. The spoken utterance is then converted to text by an LVCSR[2] system and classified using the same classification approaches as for e-mail.

6.2 Related work

The concept of using text categorization techniques for assisting agents in answering e-mail is not new. Busemann et al. (2000) developed the ICC mails system to assist call centre agents in answering e-mails by suggesting relevant solutions for incoming e-mail. They use Shallow Text Processing (STP) like word stemming, part-of-speech tagging and sentence types in combination with statistics based machine learning (SML) techniques like neural networks and support vector machines for mapping incoming mail on standard answers. Their experiments on a German e-mail corpus (containing 4,777 e-mails and 74 standard answers of which 47 used in the experiments) showed that the correct answer is selected in about 56% of the incoming e-mails using support vector machines. Neural networks and lazy learners only manage to select the correct standard answers in about 22% to 35% of

[1] IVR (Interactive Voice Response): responding on questions by pressing the buttons on a telephone
[2] Large Vocabulary Continuous Speech Recognition

the cases. Using support vector machines, the correct standard answer is selected within the top 5 results in 78% of the cases.

Gaustad and Bouma (2002) have experimented with an e-mail dataset acquired in a help desk environment in their research on Dutch text classification. Their dataset consisted of 6,000 e-mails, categorized in 69 categories (which have a standard answer assigned to it), but their experiments focused on a subset of 5,518 e-mails categorized in 69 categories. For this dataset, the results ranged from approximately 43% correct for the first suggestion of the system, to 78% correct classification in the best-5 results (the correct answer is present in the first 5 suggestions). For their experiments, Gaustad and Bouma use a Naive Bayes classifier.

6.3 Classification approach

We try to tackle the automatic answer suggestion problem by transforming it into a text classification problem. Each e-mail message is looked at as a document that should be classified and the categories in which they should be classified are the representations of the standard questions. If an e-mail is classified (i.e. mapped to a standard question), we can simply suggest the answer that is associated with the standard question representing the category. The classification of new messages is done by determining the similarity between the new messages and previously answered messages. Based on the assumption that similar questions require similar answers, the new message can then be categorized in the category that stores previously answered messages that are most similar to the new message. We state that automatically determining the similarity between new messages and previously answered messages using IR techniques outperforms the basic classification approach using manually determined keywords.

We have developed an e-mail answer suggestion system in which two classification routines can be used. We incorporated a profile based classification routine called the Rocchio classifier and an example based classification routine called the K-Nearest-Neighbour classifier. In this system, each classifier can be used using either the TF.IDF (Salton and McGill 1983) or Okapi (Robertson and Sparck Jones 1997) relevance weighting scheme.

6.3.1 Rocchio classifier

A profile based classifier is basically a classifier which embodies an explicit, or declarative, representation of the category on which it needs to take decisions. Rocchio developed an algorithm for relevance feedback for use in the vector space information retrieval model, which can be adapted to serve as a profile-based classifier. Joachims (1997) describes the use of the Rocchio Classifier using TF.IDF weights, but other weighting schemes may also be used. In the training phase, the classifier learns to classify documents by calculating a prototype vector $\overrightarrow{c_j}$ for each class C_j. In this training phase, both the normalized vectors of the positive examples for a class as well as the negative examples for a class are used. Each prototype vector is calculated as a weighted difference of the positive and negative

examples:

$$\vec{c}_j = \alpha \left(\frac{1}{|C_j|} \sum_{\vec{d}_j \in C_j} \frac{\vec{d}_j}{||\vec{d}_j||} \right) - \beta \left(\frac{1}{|D - C_j|} \sum_{\vec{d}_j \in D - C_j} \frac{\vec{d}_j}{||\vec{d}_j||} \right)$$

Where C_j is the set of training documents assigned to class j and $||\overrightarrow{d_j}||$ denotes the Euclidean length of a vector $\overrightarrow{d_j}$. Additionally, α and β are parameters that adjust the relative impact of positive and negative training examples, recommended to be 16 and 4 respectively. However, in this study the optimal parameter settings for α and β are 1 and 8 respectively, implying that the influence of negative examples should be 8 times as big as the positive examples for the best classification results. The resulting set of prototype vectors (one for each class) represents the learned model that can be used to classify a new document d' using:

$$H_{TFIDF}(d') = \arg\max_{C_j \in C} cos(\vec{c_j}, \overrightarrow{d'})$$

The classification function H_{TFIDF} (H for hypothesis) returns the category that has the highest similarity score (using the cosine function, but other similarity functions may also be used) with respect to the document to be classified. This approach can be slightly adjusted to return a ranked list (in decreasing order of similarity) of categories that are suitable for document $\overrightarrow{d_j}$ by ignoring the $\arg\max$ function and ordering the calculated similarity scores for each category in descending order (cut off at a certain threshold if pleased).

6.3.2 K-Nearest-Neighbour classifier

Example based classifiers do not build a representation for each category and do not involve in a true training phase (these classifiers are also lazy learners). A commonly used algorithm for example-based classification is the K-NN (K-Nearest-Neighbour) algorithm, implemented by Yang (1994) in the Expert System. The conditional probability that a document d_j is classified in category c_k by human judgement is given by:

$$Pr(c_k|d_j) \approx \frac{\#(assign(c_k, d_j))}{\#(d_j \in D)}$$

Where $d_1, ..., d_m$ are unique training documents and $C_1, .., C_l$ are unique categories. Furthermore, $\#(assign(c_k, d_j))$ is the number of times category c_k is assigned to document d_j and $\#(d_j \in D)$ is the number of times document d_j occurs in the document collection D. This probability is calculated since a document may have more than one occurrence in the training sample (at least after text normalization like stopword removal and stemming). Usually this equation results in a 0 or 1, indicating a category is or is not assigned to a document. The relevance score is then calculated by comparing the query q to the first K documents $d_j \in D$ using a similarity measure like the inner product or cosine, and multiplying the result with the conditional probability calculated earlier:

$$rel(c_k|q) \approx \sum_{j=0}^{K} sim(q, d_j) \times Pr(c_k|d_j)$$

Where $sim(q|D_j)$ is the similarity score calculated by the IR component and both $sim(q|d_j)$ and $rel(c_k|q)$ are scores, not probabilities. The results can be used to return a ranking (in descending order of relevance) of categories most suitable for the new document. Again, this ranking can be cut off at a certain threshold.

6.4 Natural Language Processing

E-mails have a "lower" status than traditional letters and therefore are often written sloppily: they contain a lot of spelling errors and grammatical incorrect sentences. This increases the number of words used and therefore may negatively influence the performance of the classification algorithms. To overcome (most part of) this problem, we use basic Natural Language Processing to "normalise" the e-mail before using its contents in the classification algorithms.

6.4.1 Lexical normalisation and stopword removal

The first (and most basic) step is to remove stopwords and apply some lexical normalisation to each e-mail. Lexical normalisation is nothing more than a simple process of removing all unwanted characters and strings like the sender's e-mail address or postal code and removing diacritics. Stopword removal is applied to reduce feature size and speed-up the indexing and classification process.

6.4.2 Stemming

By applying stemming we hope to improve the classification process by reducing the morphological variance of terms. If a set of documents are all about the same topic (or pose the same question in our problem), but use different morphological variants (like *swimming, swum, swam* and *swim*), a classification method is unable to relate the documents based on these terms. If we apply stemming, all documents from this set now contain the same morphological variant (i.e. *swim*) and therefore can be related. In this study we use a dictionary based stemming routine provided in the Lingware tool-kit[3]. If a word could not be found in the dictionary, the stemmer uses similar words (i.e. with the same ending and word class) for which the stemming procedure is known, and applies the same procedure to the unknown words.

6.4.3 Decompounding

Decompounding (or compound splitting) is a specific NLP routine often very useful for compounding languages like Dutch, German or Finish. By decompounding we intend to improve the classification accuracy by improving the precision and recall of the IR component of the classification system. Chen (2002) showed that decompounding can improve both recall and precision in Dutch and German IR systems. Also, Monz and De Rijke (2001) have performed successful experiments

[3]provided and implemented by Carp Technologies

on using decompounding in a Dutch IR system which caused an increase in average precision of 6.1%. Like Monz and De Rijke (2001) the Dutch lexicon of Celex is used to implement a compound splitter in our e-mail classification system.

6.4.4 Part-of-Speech tagging

Part-of-Speech (POS) tagging has proven to be very useful in IR and text categorization, mostly due to its use for disambiguation of terms. In our e-mail classification system we use POS tagging for disambiguation of terms before stemming them and as a feature selection mechanism. For instance, words from so-called open word classes carry more meaning than words from closed word classes. Kraaij and Pohlmann (1996) stated that the majority of successful query terms for an IR system in a collection of newspapers are nouns (58%), followed by verbs (29%) and adjectives (13%), while other categories are negligable. In our system we use an unsupervised transformation based tagger (provided in the Lingware toolkit).

6.4.5 Spelling correction

E-mails may contain (many) spelling errors and typo's which (for similar reasons as stemming) does not help in retrieving and classifying an e-mail. To correct (the majority of) spelling errors and typo's in our e-mails, we use a context based spelling correction routine from the Lingware toolkit, based on N-grams, Levenhstein distance and models of common made typing errors (Jurafsky and Martin 2000).

6.5 E-mail experiments

For this practical study the contact centre in question has provided a set of approximately 30,000 e-mails. Unfortunately, this corpus has not been constructed carefully for classification purposes. After removing "nonsense" e-mails (like spam, empty e-mails, error messages, etc.) and disambiguation of the categories a corpus of 16,798 e-mails categorized in 37 categories remains. The average number of e-mails per category is 454, the largest category contains 3,593 e-mails and the smallest one contains 106 e-mails. Figure 6.2 shows the distribution of e-mails per category. The results of the classification experiments are expressed in best-x classification accuracy. If the system suggests the correct answer suggestion within the top 5 of suggestions for 50% of the e-mails, the best-5 classification accuracy is 50%. We chose this best-x classification accuracy over i.e. mean reciprocal rank (MRR) to improve readability for the stakeholders of this study (contact centre managers). For comparison, we will also denote the MRR in the results section.

This study focussed on the use of IR based classification systems for suggesting relevant answers in a contact centre and Natural Language Processing to improve the classification accuracy of these systems. We conducted two series of experiments, the first series focuses on the selected classification approaches (without

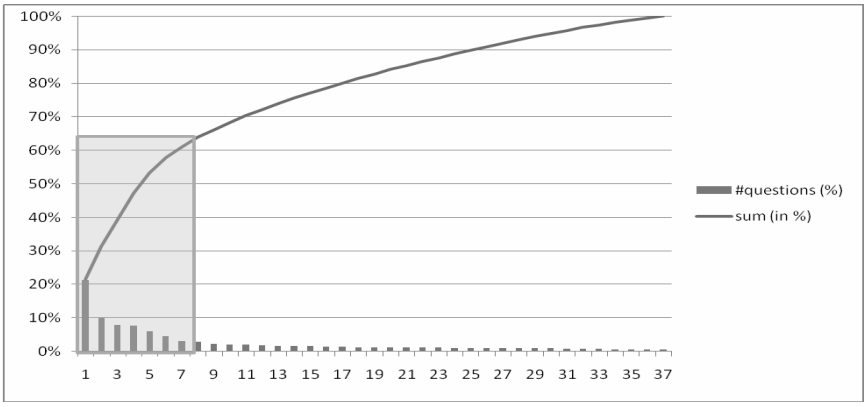

Figure 6.2: Distribution of the number of e-mails over the set of 37 categories. The largest category contains 3,593 (21%) messages, the smallest 106 (0.63%). Total amount of messages is 16798. The grey line shows the summed messages as a percentage (from 21% to 100%). The square shows the Pareto effect: the first 8 questions (21.6%) are responsible for 65% of the total amount of questions.

NLP), as the second series focuses on the use of NLP in these classification approaches. All experiments are performed using 5-fold cross validation. In table 6.5 we listed the parameter settings for our classification models. As mentioned before in section 6.3.1 the optimal parameters ($\alpha = 1$ and $\beta = 8$) for manipulating the relative influence of positive and negative examples for the Rocchio classifier differ significantly from the default parameters ($\alpha = 16$ and $\beta = 4$). The nature of the Rocchio classifier is to determine an optimal margin between the centroids of categories (the prototype vectors). The optimal margin can be found by parameter estimation that works as a feature selection procedure to find those features that are most relevant to distinguish a category from the other categories (Moschitti 2003). By increasing the influence of the negative examples with respect to the positive examples, we smoothly remove features from the positive examples that are irrelevant for distinguishing this category from the others. The relatively high influence of the negative examples in this study, shows that the set of distinguishing features for each category is relatively small and that the feature sets of the e-mail messages have an above average overlap. This means that the majority of the words used in the email messages sent to the callcenter is quite similar.

6.5.1 Experiments

In the first series of experiments we have tested two classification systems: The example based classifier (K-Nearest-Neighbour) and the profile based classifier (Rocchio). These classifiers are tested using two term relevance weighting

Table 6.1: Parameter settings for the classification models

K-NN classifier	$K = 50$
Rocchio classifier	$\alpha = 1$ and $\beta = 8$
Okapi weighting scheme	$b = 0.75$ and $k = 2$

schemes: The TF.IDF weighting scheme and the Okapi weighting scheme. Both classification approaches can use either the cosine similarity measure or the inner product similarity measure.

The second series of experiments focussed on the use of NLP within our classification approaches. We have experimented with the NLP techniques discussed in section 6.4 apart and the combination of several of these techniques together to determine the most optimal system implementation for this specific problem. In our study we found that not all NLP techniques improved the classification accuracies of both methods. For instance, decompounding caused a significant increase in accuracy for the example based classifier using Okapi weights and inner product similarity but caused a significant decrease in accuracy for the profile based classifier. In general, the example based classifier benefited more from the NLP routines than the profile based classifier.

The main results of our experiments are listed in table 6.5.1 and plotted in figure 6.3. In table 6.5.1 we printed the Mean Reciprocal Rank (MRR) and denoted the best-x classification accuracy for $x \in \{1, 3, 5\}$. In the e-mail answer suggestion system we are specifically interested in the best-5 performance, since trained contact centre agents are capable of overseeing 5 possible answers in one glance. To prove our hypothesis that de IR based classification approaches outperform the manually determined keywords based approach, we also printed the results of this keyword based approach. As a reference, the best-guess method "classifies" documents by simply suggesting the answer linked to the largest category first, the second largest second, etc.. Besides the results of our best performing classifiers, we also included the results of our best-performing classifiers combined with NLP techniques. Without NLP the example based classifier has a MRR of 0.65 and a best-5 accuracy of 84.2%, whereas the profile based classifier has a MRR of 0.61 and best-5 accuracy of 77.4%. Compared to the keywords-based approach, this is a major improvement. In the table and figure we can also see that applying NLP techniques improves the classification accuracy of the example based classifier (and the MRR increases to 0.71 due to the increase in best-1 classification accuracy), whereas the profile based classifier does not benefit that much from using NLP. Tables 6.5.1 and 6.5.1 show more detailed results of the classification experiments using NLP techniques. The example based classifier benefits most from these techniques, except from the POS-tagging/Stemming combination which causes a decrease of 10 to 12 percentage points in best-3 and best-5 classification accuracy. Using the same NLP techniques in the profile based classifier does not show any significant improvement, but rather worsened the performance

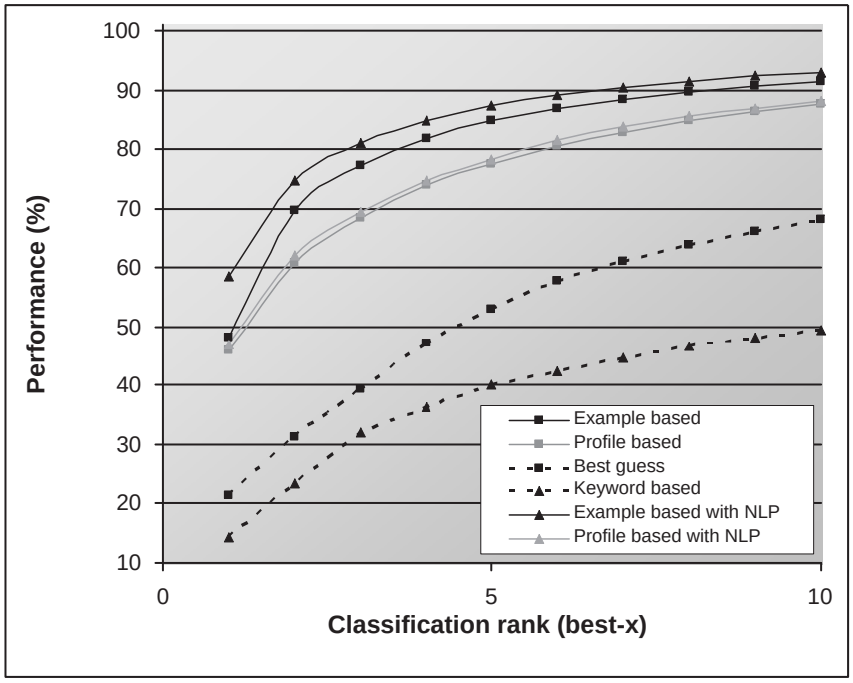

Figure 6.3: Overview of the best-x classification accuracies for the best performing example and profile based models, compared to the same models combined with NLP techniques. For comparison, the accuracy of both the best-guess approach and keywords based approach is also plotted.

of the profile based classifier (especially when decompounding was applied). Our best classification approach (Example based with NLP) is able to suggest the correct answer within a set of 5 suggestions for 87.4% of the incoming e-mails (with a MRR of 0.71).

6.6 Speech enabled call routing

As previously mentioned we also use these classification approaches in a speech enabled call routing system. Applications of speech based routing (i.e. *How may I help you?* from Gorin et al. (1997)) where a caller poses his question to a computer and is routed to for instance one of the five possible departments are widely known and implemented.

In our speech enabled call routing application, we pose a similar question and try to map de question to one of our standard questions to determine the correct action. These actions vary from playing a prompt with the most likely answer to

Table 6.2: Classification accuracies of the baseline experiments. The table lists the Mean Reciprocal Rank and best-x classification accuracies for $x \in \{1, 3, 5\}$

Approach	**MRR**	**Best-1**	**Best-3**	**Best-5**
Example based	0.65	48.2%	77.2%	84.8%
Profile based	0.61	45.9%	68.4%	77.4%
Example based with NLP	0.71	58.5%	81.0%	87.4%
Profile based with NLP	0.62	47.0%	69.5%	78.3%
Best-guess	0.36	21.4%	39.3%	53.1%
Keyword-based	0.26	14.2%	31.9%	40.0%

Table 6.3: Classification accuracies of a selection of the NLP experiments using the example based classifier. The table lists the increase or decrease in best-x classification accuracies for $x \in \{1, 3, 5\}$

NLP technique	**Best-1**	**Best-3**	**Best-5**
Example based, baseline	52.07%	74.63%	82.15%
Stopword removal	+3.58	+3.1	+2.62
Decompounding	+6.48	+6.42	+5.25
Stemming	+5.81	+5.96	+4.73
POS-tagging	+6.24	+6.01	+4.68
POS-tagging/Stemming	+3.99	-10.36	-12.79
Stopwords/Decompounding	+4.39	+4.73	+3.78
Decompounding/Stemming	+5.19	+6.24	+4.91
Stopwords/Stemming	+7.16	+6.84	-1.41
Decompounding/POS-tagging	+5.93	+6.11	+4.78
Stopwords/Decompounding/Stemming	+4.05	+4.26	+3.69

Table 6.4: Classification accuracies of a selection of the NLP experiments using the profile based classifier. The table lists the increase or decrease in best-x classification accuracies for $x \in \{1, 3, 5\}$

NLP technique	**Best-1**	**Best-3**	**Best-5**
Profile based, baseline	45.87%	68.45%	77.40%
Stopword removal	-1.77	-0.78	-0.02
Decompounding	-4.11	-3.72	-2.69
Stemming	+0.36	-0.24	-0.56
POS-tagging	+1.16	+1.06	+0.91
POS-tagging/Stemming	-2.28	-1.85	-1.55
Stopwords/Decompounding	-5.96	-5.01	-3.87
Decompounding/Stemming	-3.41	-3.41	-3.34
Stopwords/Stemming	-3.36	-0.664	-0.05
Decompounding/POS-tagging	-3.36	-2.33	-1.58
Stopwords/Decompounding/Stemming	-7.19	-5.06	-3.86

routing to a self-service application or specific department of the company. This type of application brings an extra speech recognition task to the application. The LVCSR results of each spoken utterance are sent to the classification system and a ranked list with best matching standard questions is returned. The caller can then chose the best matching standard question to proceed in the application.

6.6.1 Corpus

To determine the classification accuracies of these classification systems if instead of written text, recognized speech is used, we have collected a set of 3,322 spoken utterances in our speech enabled call routing application implemented in a Dutch contact centre of a telecom provider. The recorded utterances can be categorized in 36 categories with an average category size of 92. The smallest category contains only 3 utterances as the largest category contains 279 utterances. The size of this corpus is a bit small for the intended experiments, but since all utterances have to be manually transcribed (in order to study the effect of speech recognition) and categorized for training and testing, no more data was available at the current time.

6.6.2 Experiments

The experimental set-up is similar to that of the e-mail answer suggestion experiments. We focus on the example based classification approach (K-Nearest-Neighbour with $K = 25$) with the Okapi weighting scheme ($b = 0.75$ and $k = 2$) using inner product similarity and no additional NLP techniques other than stopwordremoval. For the LVCSR we use a commercially available speech recognizer designed for dictation. Within this recognizer we use the 'Unisex' acoustic model

Table 6.5: Classification accuracies of the speech recognition classification experiments. The table lists the Word Error Rate, Mean Reciprocal Rank and best-x classification accuracies for $x \in \{1, 3, 5\}$

Approach	**WER**	**MRR**	**Best-1**	**Best-3**	**Best-5**
Transcriptions	0%	0.79	69.5%	87.6%	91.9%
General context	74%	0.48	38.1%	53.3%	59.1%
Website context	67%	0.58	48.0%	64.5%	70.1%
Transcription context	55%	0.67	57.1%	75.0%	80.5%
Best-guess	-	0.21	8.4%	20.1%	30.0%

because the caller identity and gender are unknown (before we recognize the utterance). We perform experiments with four different context models, each trained with a specific type of documents:

- A general context with additional training of CGN data (telephone and face-to-face conversations (Oostdijk 1999)): 'General context'
- The context above, additionally trained with relevant context information of the telecom provider (taken from the company's website: 'Website context'
- The context above, additionally trained with transcriptions of spoken utterances in the speech enabled IVR application: 'Transcription context'

For comparison, we perform our experiments by classifying the speech recognized texts and also the orthographic transcriptions to study the influence of the speech recognition induced errors on our classification accuracy.

The results of our experiments are listed in table 6.6.2 and plotted in figure 6.4. For each of the experiments we denoted the MRR, best-x classification accuracy for $x \in \{1, 3, 5\}$ to reflect the classification performance and the Word Error Rate to reflect the speech recognition performance.

The WER (Word Error Rate) of the recognized utterance is pretty high, mostly because we are forced to use an unisex acoustic model and have to deal with noisy telephone speech. However, if we apply a well trained context (language model), the WER decreases to 55% and a best-5 classification accuracy increases to over 80%. Unfortunately, this is 10% less than the accuracy we could have yielded if there were no speech recognition induced errors. The next goal would be to improve the speech recognition component in order to decrease the WER; this can be done by adjusting the language models with more transcriptions. Moreover, if we manage to incorporate a gender detection routine, we can apply gender-specific acoustic models in the speech recognition task. The classification accuracies are also expected to increase if we expand the size of the training set: In these series of experiments we were forced to use just 3,300 examples instead of the almost 17,000 examples for the e-mail experiments, while the number of categories are almost equal (36 for speech opposed to 37 for e-mail).

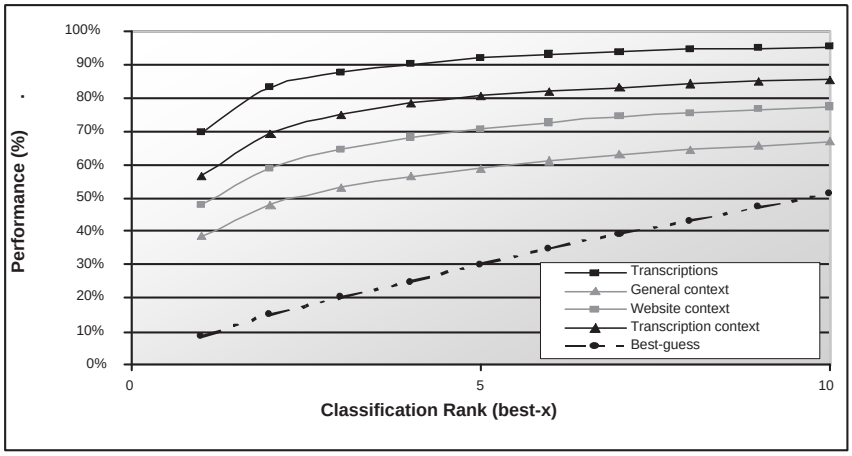

Figure 6.4: Overview of the best-x classification accuracies for the various context models (General, Website, Transcription) and full hand made transcription. For comparison, the accuracy of the best-guess approach is also plotted. The figures are based on the spoken Telecom corpus.

6.7 Conclusions and future work

In our introduction we stated that IR based classification would outperform the keyword based classification approach in our e-mail answer suggestion problem and that the use of Natural Language Processing would even further improve the accuracy. We showed that by using IR based classification approaches, the best-5 classification accuracy more than doubled from 40% to approximately 85%, meaning that for almost 85% of the incoming e-mails, the correct answer suggestion is listed within a ranked list of 5 possible answer suggestions. If we apply Natural Language Processing within the classification task, the best-5 classification accuracy rises to more than 87%. A relatively small increase, but if we focus on the best-1 classification accuracy, the increase is more than 10%. In conclusion we developed an e-mail answer suggestion system that suggests the correct answers within a list of 5 possible suggestions in 87% of the times and, moreover, places the correct answer suggestion at the top of this list in almost 60% of the cases. Furthermore we showed that these classification approaches are also well suitable in speech enabled call routing systems where callers respond on the question: ”How may we help you?”.

Our future work will focus on the improvement of the speech enabled call routing applications. We intend to boost the best-3 classification accuracy over 80% by improving the speech recognition results and better matching of these results with the classification models. To improve the classification accuracy, we may also ben-

efit from NLP techniques to find a better match between the classification models and the speech recognized utterances (e.g. by using feature expansion by adding frequent confusion words). In order to provide reliable confidence information to the classification results, we will also focus on the use of other classification approaches. If we are confident that a standard question is the best match for the spoken utterance of the caller, we can improve the self-service level by providing the correct answer immediately instead of prompting the caller to select his question from a set of relevant questions.

References

Busemann, S., S. Schmeier, and R.G. Arens (2000), Message classification in the call center, *Proceedings of the sixth conference on Applied natural language processing*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 158–165.

Chen, A. (2002), Cross-language retrieval experiments at clef 2002, *Proceedings of CLEF-2002*, pp. 28–48.

Gaustad, T. and G. Bouma (2002), Accurate stemming of dutch for text classification, *Language and Computers* **45** (1), pp. 104–107.

Gorin, A. L., G. Riccardi, and J. H. Wright (1997), How may I help you?, *Speech Communication* **23** (1/2), pp. 113–127.

Joachims, T. (1997), A probabilistic analysis of the rocchio algorithm with tfidf for text categorization, *in* Fisher, Douglas H., editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, US, Nashville, US.

Jurafsky, D. and J.H. Martin (2000), *Speech and Language Processing*, Prentice Hall, New Jersey.

Kraaij, W. and R. Pohlmann (1996), Viewing stemming as recall enhancement, *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, New York, USA, pp. 40–48.

Monz, C. and M. De Rijke (2001), Shallow morphological analysis in monolingual information retrieval for dutch, german and italian, *Proceedings CLEF 2001*, Springer Verlag, pp. 262–277.

Moschitti, A. (2003), A Study on Optimal Parameter Tuning for Rocchio Text Classifier, *LECTURE NOTES IN COMPUTER SCIENCE* pp. 420–435, Springer.

Oostdijk, Nelleke (1999), Building a corpus of spoken dutch, *Computational Linguistics in the Netherlands 1999, Selected Papers from the Tenth CLIN Meeting, December 10, OTS Utrecht*.

Reed, William J. (2001), The pareto, zipf and other power laws, *Economics Letters* **74** (1), pp. 15–19.

Robertson, S.E. and K. Sparck Jones (1997), Simple, proven approaches to text

retrieval, *Technical report*, City University London and University of Cambridge.

Salton, G. and M.J. McGill (1983), *An introduction to Modern Information Retrieval*, McGraw-Hill.

Yang, Y. (1994), Expert network: Effective and efficient learning from human decisions in text categorisation and retrieval, *in* Croft, Bruce W. and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Springer-Verlag, pp. 13–22.

7

Automatic Synonymy Extraction

A Comparison of Syntactic Context Models

Kris Heylen, Yves Peirsman and Dirk Geeraerts
QLVL - University of Leuven, Belgium[1]

Abstract

Distributional models of lexical semantics identify semantically similar words through contextual similarity. Previous studies have shown that syntactic contexts are especially good at finding (near) synonyms. In this paper, we compare models based on eight different syntactic dependency relations and we evaluate their separate and combined performance on a test set of Dutch nouns. Firstly, we analyze to what extent their results overlap. Secondly, we assess the overall performance of the models by looking at the average similarity of the words they return. And thirdly, we compare the specific semantic relations retrieved by the models. The analyses show that although models based on the subject and object relation give the most consistent results, it is the model based on adjective modification that gives the best results. It even outperforms the combined model at finding true synonyms.

7.1 Introduction

The automatic retrieval of semantically similar words has become an important task in NLP research with applications in Information Retrieval, Word Sense Dis-

[1] Yves Peirsman is a Ph.D. Fellow of the Research Foundation - Flanders (FWO)

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 101–116
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: kris.heylen@arts.kuleuven.be

ambiguation, Thesaurus Extraction, Anaphora Resolution and even Parsing. So-called distributional models of lexical semantics have turned out to be one of the most promising approaches for modelling semantic similarity. They rely on the assumption that words with a similar meaning tend to occur in similar contexts and, consequently, that a word's meaning can be modelled as a function of the contexts it occurs in. In practice, these models gather statistics about the co-occurrence of a word with a large number of context features from corpora and put these into a vector. The semantic similarity between two words is then measured as the distributional similarity between their respective context vectors.

Over the last decade, various implementations of the distributional approach have been developed. They mainly differ with respect to the restrictions they impose on what counts as *context* for modelling the meaning of a target word. Some variants allow all the other words in the same document, others only take words in a predefined window around the target word into account, and still others use only context words in a specific syntactic relation to the target word. It is clear that these different types of context features are likely to capture different kinds of semantic information. However, until recently, little was known about the influence of the context definition on the semantic information present in these word vector spaces. While most researchers choose one specific model and apply it to their task, "comparisons between the ... models have been few and far between in the literature" (Padó, Sebastian and Lapata, Mirella 2007). Yet, without any knowledge of the linguistic characteristics of the models, it is impossible to know which approach is best suited for a particular task, and why. In previous studies (Yves Peirsman and Kris Heylen and Dirk Speelman 2007, Yves Peirsman and Kris Heylen and Dirk Speelman 2008, Kris Heylen and Yves Peirsman and Dirk Geeraerts and Dirk Speelman 2008), we compared the performance of models using a pre-defined context window and those relying on syntactically related words. These studies showed that the syntactic model with its very strict context definition outperformed the other models in finding semantically similar nouns for Dutch. That syntactic model was based on eight syntactic dependency relations. However, our evaluation did not differentiate between these eight relations. In this paper, we will focus on models that are based on each of the eight syntactic relations separately. On the basis of a test set of Dutch nouns, we will compare the models both among each other, as well as with the combined model. This way, we want to find out which relation is most informative for detecting semantic similarity and contributes most to the overall success of the syntactic model. More specifically, we will first investigate to what extent the results of the different models overlap. Then we analyse the overall quality of the results in terms of their average semantic similarity to the test set, and finally, we will assess how well the models can detect specific semantic relations like synonymy.

In Section 7.2, we first situate our approach in the broader field of research into distributional models and we discuss previous studies into the properties of syntactic models. Section 7.3 discusses the data and parameter settings we used in our test set-up. Section 7.4 first presents the evaluation scheme we applied and then discusses, consecutively, the overlap of the models, their overall performance

and their ability to detect specific semantic relations. Finally, in section 7.5, we wrap up with conclusions and some suggestions for future research.

7.2 Related work

7.2.1 Distributional models

Distributional models, a.k.a. word space models, semantic space models or vector-based models of lexical semantics, rely on insights that were already formulated in the 1950's by Zelig Sabbettai Harris (1954), Warren Weaver (1955) and John Rupert Firth (1957), viz. that a word's meaning can be induced from the contexts it appears in. However, it was not before the 1990's and the advent of large corpora and increased computational power that a practical implementation became feasible. Distributional models actually became popular first within cognitive psychology as a way to model lexical learning and word memory (Kevin Lund and Curt Burgess 1996, Thomas K. Landauer and Susan T. Dumais 1997, Will Lowe and Scott McDonald 2000) but were soon enthusiastically adopted by the NLP community.

Although all word space models are based on the same underlying assumption, they do come in many different flavours. The main difference between them lies in how they define the central notion of *context*. Figure 7.1 offers a classification according to context definition. Document-based models use whole documents or paragraphs as contexts so that words that often co-occur in documents appear as semantically similar. Latent Semantic Analysis (Thomas K. Landauer and Susan T. Dumais 1997) is probably the best known example of this type. Because they use documents as context, they are especially useful for the grouping of topic identifying terms in document classification.

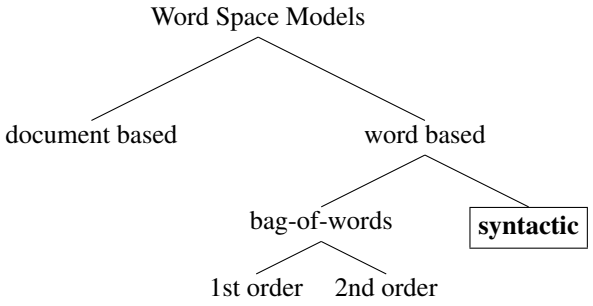

Figure 7.1: Syntactic models within the family of Distributional Models

For extracting tight semantic relations like synonymy, however, word-based models have turned out to be more suitable. They restrict contexts to the words in near proximity to the target word and treat two words as similar if these often co-occur with the same context words. Unlike document-based models, they do not expect target words to co-occur regularly with each other. Within the class of

word-based models, we can make a distinction between bag-of-word and syntactic models. Bag-of-word models simply look at the context words that appear in a pre-defined window around the target word. They are called bag-of-words models because they do not differentiate between the words within the context window. Context words with different POS values or syntactic functions are all treated on a par. A further subdivision can be made between first-order and second-order bag-of-words models. In the case of first-order models, the context features are the words that directly co-occur with the target word in the pre-defined window (Joseph P. Levy and John A. Bullinaria 2001), whereas second order bag-of-word models make use of the second order co-occurrences, i.e. the context words of the first order co-occurrences. By doing so, they should allow to generalize over meaning related context words and avoid data sparseness. These properties make that second-order models have been primarily used in Word Sense Disambiguation for grouping word tokens in sense clusters (Hinrich Schütze 1998), rather than for the clustering of word types.

Finally, we come to the syntactic or dependency-based models. Unlike the bag-of-word models they do impose a specific relation between the target and its context words. They only take context words into account that stand in a pre-defined syntactic dependency relation to the target word. Context features are then words like verbs governing the target word in its subject function or adjectives modifying the target, plus the respective dependency relation. These models have proven to be especially apt at finding words with tightly related meanings and they will be the focus of this study.

7.2.2 Syntactic models

Syntactic models owe their success to the fact that not all context words are equally informative for inferring a word's meaning. For example, verbs subcategorize for subject or object nouns from specific semantic classes. Consequently, a subject or object noun's governing verb tells a lot more about that noun's meaning than another randomly chosen context word. The same holds for adjectives modifying a given noun. Researchers have tried to capitalize on this link between semantics and morpho-syntactic structure to various degrees by taking into account part of speech tags (Dominic Widdows 2003, Klaus Rothenhäusler and Hinrich Schütze 2007), shallow syntactic analyses (Gregory Grefenstette 1994, Lillian Lee 1999, James R. Curran and Marc Moens 2002) and full syntactic parses (Dekang Lin 1998). Only the latter approach fully exploits the semantic information contained within syntactic dependency relations but also requires most resources. However, thanks to the advent of robust automatic parsers, the use of full-blown dependency information has become a viable option that we will also pursue here.

In their excellent overview article, Padó, Sebastian and Lapata, Mirella (2007) propose a general framework for implementing distributional models that integrates the additional parameters necessary to describe dependency-based models. One of those parameters is the *context selection function*, which specifies the relation that must hold between target and context words. For bag-of-words models,

this relation is simply one of occurrence within the context window. In the case of syntactic models, it is a bit more complex. The relation is then stated in terms of the possible paths through a syntactic dependency tree that can connect a context word with a target word. These paths can be of length one, e.g. between a target noun and its governing context verb, but they can also be longer. In the NP "the man with the hat", the context word *hat* is a postmodifier to the target noun *man* linked by a path of length two with connections from *man* to *with* and from *with* to *hat*. If the dependency tree is regarded as a graph with the words as nodes and the dependency relations as edges, the context selection function specifies, what length a dependency path can have, which parts-of-speech the start, end and intermediate nodes are allowed to be, and of what type the edges can be. It is this context selection parameter that we will vary in our study, while keeping the other parameters constant.

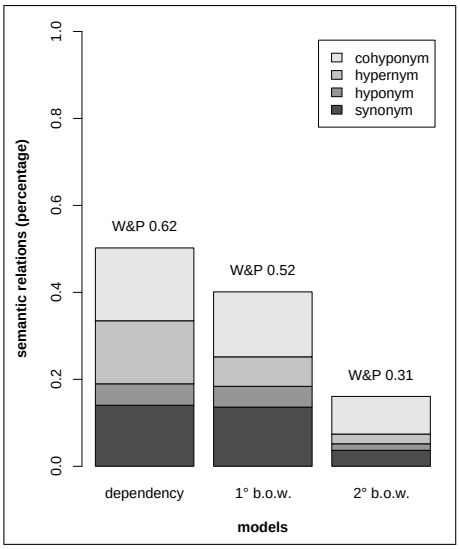

Figure 7.2: Performance of the syntactic model compared to the first and second order bag-of-words models (Wu & Palmer score and semantic relations for single nearest neighbours)

For Dutch, distributional models have been applied to Semantic Class Induction (Van de Cruys, Tim 2005), Multi Word Expression Extraction (Van de Cruys, Tim and Villada Moirón, Begoña 2007), Word Sense Discrimination (Van de Cruys, Tim 2007), Cognitive Word Association Modelling (De Deyne, Simon and Storms, Gert 2008) and Question Answering (Lonneke van der Plas and Gosse Bouma 2005a). Van der Plas and Tiedemann have compared the performance of distributional models based on monolingual and parallel corpora (Lonneke van der Plas and Jörg Tiedemann 2006). Our own studies (Yves Peirsman and Kris Heylen and Dirk Speelman 2007, Yves Peirsman and Kris Heylen and Dirk Speelman 2008, Kris Heylen and Yves Peirsman and Dirk Geeraerts and Dirk

Speelman 2008) have compared bag-of-word and syntactic models, and found, as Figure 7.2 shows, that a syntactic model outperforms the bag-of-word models both in terms of overall performance and specific semantic relations retrieved[2]. As the best performing model, we will focus on the syntactic model here. In this regard, our study links up directly with previous work by van der Plas and Bouma (Lonneke van der Plas and Gosse Bouma 2005b). They too compared the overall performance of syntactic models based on six different dependency relations and section 7.4.2 will partially be a replication of their experiments. However, this study will look at two additional dependency relations and will not only assess overall performance, but also the overlap of the models and the specific syntactic relations they retrieve.

Table 7.1: Paths and examples for the eight dependency relations

Rel.	Path	Example
su	noun $\overset{su}{\rightarrow}$ verb	Het *meisje* slaapt (The girl sleeps)
obj	noun $\overset{obj1}{\rightarrow}$ verb	Hij eet een *appel* (He eats an apple)
pc	noun $\overset{obj1}{\rightarrow}$ prepositon $\overset{pc}{\rightarrow}$ verb	Ze luistert naar de *radio* (She listens to the radio)
advPP	noun $\overset{obj1}{\rightarrow}$ preposition $\overset{mod}{\rightarrow}$ verb	Hij woont in een *dorp* (He lives in a village)
pmPP	noun $\overset{mod}{\leftarrow}$ preposition $\overset{obj1}{\leftarrow}$ noun	Het *meisje* met de jurk (The girl with the dress)
adj	noun $\overset{mod}{\leftarrow}$ adjective	De gelaarsde *kat* (The booted cat)
app	noun $\overset{app}{\leftarrow}$ noun	De *koningin*, een wijze vrouw (The queen, a wise woman)
cnj	noun $\overset{cnj}{\leftrightarrow}$ noun	De *krekel* en de mier (The cricket and the ant)

7.3 Set-up

The data for our experiments consists of the 300 million word Twente Nieuws Corpus of Dutch lemmatised and parsed newspaper text (Roeland J.F. Ordelman 2002)[3]. Parsing was done at the University of Groningen with the Alpino dependency parser for Dutch (Gertjan van Noord 2006). Based on Alpino's parsing scheme, we selected eight types of syntactic dependency relations for constructing our word spaces. The target word's function in these relations was one of the following:

1. **su**: subject of verb v
2. **obj**: direct object[4] of verb v
3. **pc**: prepositional complement of verb v introduced by preposition p

[2] See section 7.4 for a description of these evaluation measures
[3] Publication years 1999 up to 2002 of *Algemeen Dagblad*, *NRC*, *Parool*, *Trouw* and *Volkskrant*
[4] This includes the subjects of passive verbs

4. **advPP**: head of an adverbial PP of verb v introduced by preposition p
5. **pmPP**: postmodified by a PP with head n, introduced by preposition p
6. **adj**: modified by adjective a
7. **app**: modified by an apposition with head n
8. **cnj**: coordinated (via a conjunctor) with head n

Each specific instantiation of the variables v, p, a, or n led to a new context feature. Table 7.1 shows the dependency paths as they were extracted from Alpino's output along with some examples. In su, obj and pc, the target noun is the dependent word in the relation, and in cnj there is a symmetrical relation between the target and context word. In the other cases, the target word is the head. Three relations (pc, advPP, pmPP) have a path length of two. The others are direct relations of path length one. We extracted from the lemmatised corpus the 10,000 most frequent nouns and recorded for each of them with which specific instantiations of the eight relation types they occurred and how often. Based on this information, we constructed nine word spaces: one for each of the eight depency relations separately, and one for the combination of relations. For the models based on separate dependency relations, only the 1000 most frequent features were used to guarantee computational feasability. For the combined model, we kept the original dimensionality from our previous studies and took the 4000 most frequent features into account[5]. The remaining parameters were kept constant and were set as follows:

Weighting scheme Context vectors contained the point-wise mutual information (Kenneth Ward Church and Patrick Hanks 1990) between the feature and the target, rather than raw frequency.

Similarity metric The cosine of the angle described by two context vectors was used to measure the similarity between these vectors.

Using these models, we calculated for each target noun the 100 most similar nouns among the remaining 9999 possibilities, which we will designate as the target's *nearest neighbours*. To give an idea of the output, Table 7.2 gives the first nearest neighbour found by the nine models for a random sample of 10 target nouns.

7.4 Results and Discussion

We performed three types of evaluation: overlap between the models, overall performance in terms of the nearest neighbours' average semantic similarity to their targets, and specific semantic relations retrieved. The two latter types of evaluation use a gold standard, which we will discuss briefly first.

We evaluate our models against the Dutch part of EuroWordNet (Vossen, Piek 1998). Like its English counterpart, EuroWordNet is a lexical database structured

[5]Distribution over relations: adj 843; advPP 431; app 57; cnj 524; obj 675; pc 336; pmPP 412; su 684

Table 7.2: Random sample of target words and their nearest neighbours

TARGET	adj	advPP	app	cnj	all
aanzien	uitstraling	aanleiding	vliegverkeer	prestige	prestige
capriool	frats	steunpilaar	beschadiging	ommekeer	stunt
flirt	uitstapje	oogst	azijn	drank	avontuur
grondlegger	opsteller	spil	coup	oprichter	oprichter
grot	kelder	kelder	voetbalveld	ravijn	dorp
kerkgebouw	kerk	kerk	zuivel	synagoge	kerk
opdrachtgever	financier	werkgever	aandeelhouder	ontwerper	klant
thee	koffie	koffie	zwijn	koffie	koffie
toilet	wc	strand	bushalte	douche	wc
trots	held	stelligheid	maatschappij	liefde	vreugde

TARGET	obj1	pc	pmPP	su	all
aanzien	prestige	populariteit	imago	populariteit	prestige
capriool	frats	ophef	zonneschijn	bijval	stunt
flirt	hoogmoed	provocatie	omgang	rechtszaak	avontuur
grondlegger	schilder	exponent	geschiedenis	schepper	oprichter
grot	paleis	restaurant	gehucht	hiernamaals	dorp
kerkgebouw	pand	autosnelweg	supermarkt	hangar	kerk
opdrachtgever	eigenaar	werkgever	uitvoerder	werkgever	klant
thee	koffie	koffie	koffie	koffie	koffie
toilet	wc	wc	wc	keuken	wc
trots	zelfvertrouwen	ontzag	imago	opluchting	vreugde

as a hierarchical network of concepts, each represented as the set of synonyms (synset) that refers to it. The Dutch section of EuroWordNet contains 44K synsets, which is a fair bit sparser than English WordNet (117K synsets). Our evaluations are based on the target-neighbour pairs retrieved by the models and, of course, only pairs with both the target and nearest neighbour present in EuroWordNet can be assessed. To allow for a fair comparison, we wanted to evaluate all the models on exactly the same set of target words. Therefore, we could only take target words into account that were themselves present in EuroWordNet, and for which each of the first nearest neighbours retrieved by the 9 models was also present. Because of the relative sparseness of EuroWordNet, this led to a drastic reduction of the data set to 2749 target words. Needless to say, this means the results below should be interpreted with some caution.

7.4.1 Overlap

As a first evaluation, we want to know to what extent the models retrieve the same related words for our test set of Dutch nouns. To do so, we use the overlap metric developed by Sahlgren (Magnus Sahlgren 2006). The metric reflects how similar the results of two models are, simply by calculating the overlap between the nearest neighbours found for each target word[6]: For each of the 10,000 nouns in

[6]This overlap is generally very low. Sahlgren, for instance, found a maximum of around 10% overlap between the document-based and bag-of-word models.

our test set[7], we took the 100 nearest neighbours and then calculated how many neighbours the models shared. The total number of shared neighbours divided by 10,000 then gives an average overlap between each of the models expressed in percentages[8]. For the combined model, the resulting similarity matrix showed, unsurprisingly, that the degree of overlap with separate relation models follows closely the relative frequency of these relations in the dimensions of the combined model. The real interesting question is to what extent the separate relation models, based on different information, overlap. We therefore took the similarity matrix for these models, turned it into a dissimilarity matrix by taking $1 - overlap$, and fed this into a hierarchical cluster analysis using complete linkage as a cluster criterium. Figure 7.3 shows that the results of the models based on the direct object and subject relation are most similar with an overlap of 24% (and thus a distance of 0.76). This overlap is actually quite high and might be explained by the fact that in both relations the target noun is directly dependent on a context verb. The next subcluster consists of the adjective and coordination-based model. These two relations do not have a lot in common that might explain their clustering. Such a communality does exist for the next subcluster consisting of the prepositional complement and adverbial PP models. Both relations connect the target noun to a governing context verb via a prepostion in a two step dependency path. In fact, the distinction between prepositional complements and adverbial PP's is sometimes arbitrary in Alpino's parsing scheme. For example, *radio* in *luisteren naar de radio* (listen to the radio) is parsed as a prepositional complement, whereas *televisie* in *kijken naar de televisie* (watch television) is regarded as an adverbial PP. Finally, the dendrogram shows that the nearest neighbours retrieved by the post-modifying and apposition models overlap least with those of the other models. We now know how similar the results of the different models are. However, this doesn't tell us anything about the quality of these results. That is what we will look at in the next two sections.

7.4.2 Overall performance

Following previous studies (Lonneke van der Plas and Gosse Bouma 2005b, Van de Cruys, Tim 2006, Yves Peirsman and Kris Heylen and Dirk Speelman 2007) we analyse the overall performance of our models by measuring the average semantic similarity of the nearest neighbours to their targets as recorded in EuroWordNet. To do so, we use the Wu & Palmer similarity score (Zhibiao Wu and Martha Palmer 1994) which has become somewhat of a standard for measuring similarity in lexical taxonomies. Wu & Palmer basically measure how far two words are removed in the hierarchy and apply a normalization for the relative hierarchy depth, which then results in a score ranging from 0 (no similarity) to 1 (perfect similarity) [9].

[7]Since we do not have to rely on EuroWordNet here, we can use the complete data set.

[8]Note that this overlap measure does not take into account the rank of nearest neighbours.

[9]If a word occurs at different places in the hierarchy because of polysemy, only the highest Wu & Palmer score is taken into account. When the system returns, say, *depository* for a polysemous word

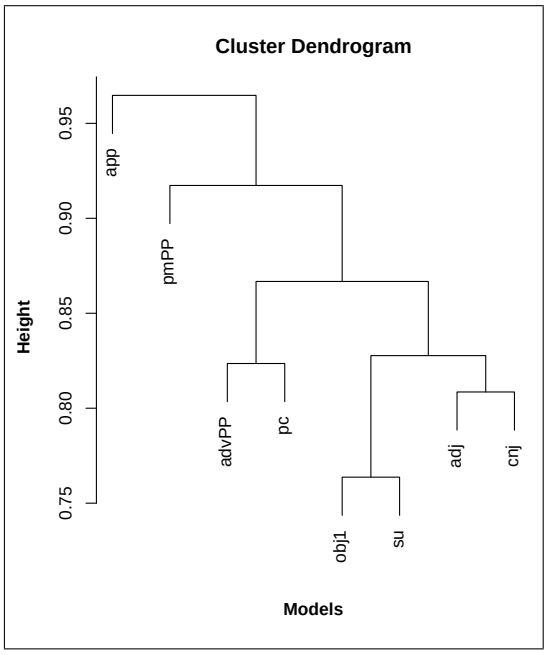

Figure 7.3: Clustering of the models based on their results

Figure 7.4 shows this average similarity for the 1st, 10, 50 and 100 most related words retrieved by the models[10]. We see that the combined model performs best with an average Wu & Palmer score of 0.65 for the first nearest neighbour. This shows that it is worthwhile to combine different dependency relations. With two relations extra (advPP and pmPP), we do even slightly better than van der Plas and Bouma's (Lonneke van der Plas and Gosse Bouma 2005b) maximum score of 0.60 on their test set, although that might also be due to the bigger size of our training corpus. Note that our adjective model on its own also scores a surprising 0.61, which is almost as high as the combined model. This result confirms that adjectives are highly informative for modelling the semantics of the nouns they modify. The adjective model is closely followed by the one based on the direct-object relation with a 0.59 score. The four next models (su, pc, cnj, advPP) perform a bit less well with scores around 0.50 for the first nearest neighbour, but this difference gets less pronounced when more nearest neighbours are taken into account. The worst models are clearly those based on the post-modifying PP and apposition relation. The pmPP model still scores reasonably well for the first nearest neighbour, but

like *bank*, it seems fair to assume that the identified similarity is to the financial meaning of bank rather than to the river side meaning.

[10]As explained in section 7.4 we required that at least the 1st nearest neighbour was present in EuroWordNet for all models. For nearest neighbours up to rank 100, we only took those into account that were present in the database and simply ignored the others for calculating the average similarity.

then deteriorates quickly. The apposition model performs rightout poorly across the whole spectrum with an average similarity as low as 0.23 for the first nearest neighbour.

Interestingly, the ranking of the models based on average similarity does not completely correspond to their relative degree of overlap: the subject and object models gave the most consistent results according to the overlap measure, but it is actually the adjective model that performs best when evaluated against a gold standard. On the other hand, the poor performance of the apposition model is reflected in its low overlap with the other models. Finally, we also point out that our results do not completely match those found by van der Plas and Bouma (Lonneke van der Plas and Gosse Bouma 2005b). In their experiment, the object-based model performed best, although the adjective model came in second. The main difference lies in the performance of the apposition model, which still scored a fairly good 0.51 in van der Plas and Bouma's experiment. These differences might be due to the fact that a different test set was used.

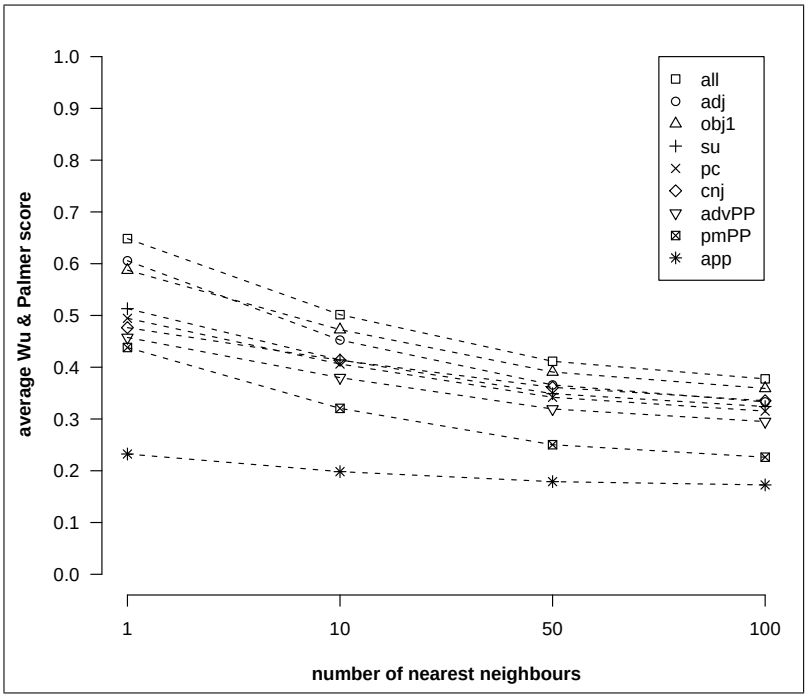

Figure 7.4: Overall performance of the 9 models

7.4.3 Specific semantic relations

Although the average semantic similarity of the nearest neighbours might give a good idea of overall performance, many applications of distributional models, like thesaurus extraction or query expansion, are foremost interested in finding specific semantic relations. To evaluate the models' performance on this task, we checked which semantic relation, if any, each first nearest neighbour entertained with its target according to EuroWordNet. Four semantic relations were taken into account (in decreasing order of semantic relatedness): synonymy, hyponymy, hypernymy and co-hyponymy. They were defined as follows:

synonym word occurring in the same synset as the target

hyponym word occurring in a synset that is a direct daughter of the target's synset

hypernym word occurring in a synset that is a direct mother of the target's synset

cohyponym word occurring in a synset that is a direct daughter of the target's hypernymic synset

Note that we use a strict definition of semantic relatedness by only allowing a minimal number of steps between a target and its neighbour in the hierarchy[11].

Figure 7.5 shows the relative frequency of the four semantic relations among the most related words retrieved by the nine models. The aggregated percentages confirm by and large the results of the overall performance analysis: the combined model performs best with 55% of its results displaying one of the four semantic relations. The adjective and object model follow at 48% and 45% respectively. The apposition-based model performs worst with only 10% of the retrieved words having a semantic relation to the target. However, if we look at the individual semantic relations, we do see interesting differences between the models that were not revealed by the overall performance analysis. First of all, the adjective model is slightly better than the combined model at finding true synonyms (15.4% versus 14.7%). Secondly, in the group of average performing models, we see that the model based on post-modifying PP's is also notably better at finding true synonyms than its overall performance score would predict. Both observations suggest that modifiers are clearly good indicators of a noun's precise meaning, resulting in a better retrieval of tightly related words. If strict synonymy for nouns is what is needed for an application, distributional models based on adjective modification alone might be the best option to go with.

7.5 Conclusions and future work

Dependency-based distributional models were already known to perform quite well on the task of finding semantically related words when compared to bag-of-word competitors. Our detailed analysis of dependency-based models in this

[11] As with the Wu & Palmer score, only the shortest connection in the hierarchy was taken into account for target-neighbour pairs with multiple connections due to polysemy.

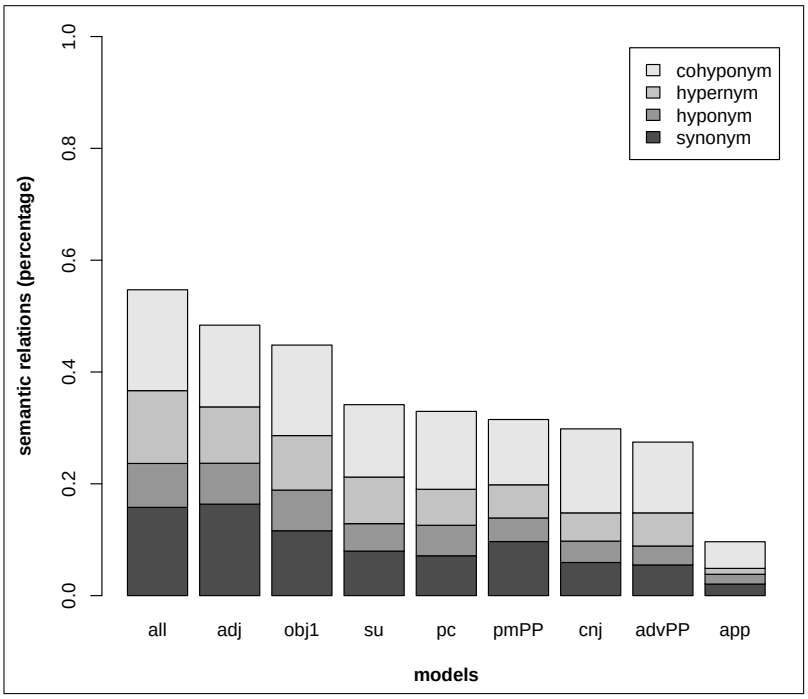

Figure 7.5: Semantic relations retrieved by the 9 models

paper has shed more light on the question why these models are more successful. More specifically, we were able to tease out their sources of information for noun semantics by comparing models that used specific types of dependency relations as context features. Our three evaluation schemes were able to highlight different aspects of their performance. By comparing the overlap in results, we saw that models with context features based on direct verb dependency, i.e. object and subject relations, give the most consistent results. However, our overall performance analysis showed that these are not necessarily the best results: those were generated by the model using adjectives as context features. The evaluation of overall performance also showed that a combination of dependency relations allows to retrieve words with a slightly higher average similarity than the best performing individual relation. Finally, the analysis of specific semantic relations retrieved showed that models based on modifiers, either adjectives or post-modifying PP's, are especially good at finding true synonyms. The adjective-based model even outperformed the combined model on this task.

We have only looked at eight types of dependency relations so far and in a next step we would like to extend this set. In particular, an analysis of relations with longer path lengths could reveal whether post-modification by certain relative clauses, say, those headed by specific verbs is also informative for noun seman-

tics. Apart from analysing which dependency relation works best individually, we would also like to find out which combination is best suited to retrieve semantically similar words. We have looked at one combination, but for eight relations there are 248 possible combinations. By studying how individual dependency relations interact, we want to get a systematic overview of the links between a word's meaning and the syntactic contexts it occurs in.

References

De Deyne, Simon and Storms, Gert (2008), Word associations: Network and semantic properties, *Behavior Research Methods* **40** (1), pp. 213–231.

Dekang Lin (1998), Automatic retrieval and clustering of similar words, *Proceedings of the 17th international conference on Computational linguistics*, pp. 768–774.

Dominic Widdows (2003), Unsupervised methods for developing taxonomies by combining syntactic and statistical information, *Proceedings of the Joint Human Language Technology Conference and Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 197–204.

Gertjan van Noord (2006), At Last Parsing Is Now Operational, *in* Piet Mertens and Cedrick Fairon and Anne Dister and Patrick Watrin, editor, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*.

Gregory Grefenstette (1994), Corpus-derived first, second and third-order word affinities, *Proceedings of the Sixth EURALEX International Congres*.

Hinrich Schütze (1998), Automatic word sense discrimination, *Computational Linguistics* **24** (1), pp. 97–124.

James R. Curran and Marc Moens (2002), Scaling Context Space, *Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL)*, pp. 231–238.

John Rupert Firth (1957), A synopsis of linguistic theory 1930-1955, *Studies in Linguistic Analysis*, Oxford Philological Society, pp. 1–32.

Joseph P. Levy and John A. Bullinaria (2001), Learning Lexical Properties from Word Usage Patterns: Which Context Words Should be Used?, *in* R.M. French and J.P. Sougne, editor, *Connectionist Models of Learning, Development and Evolution: Proceedings of the Sixth Neural Computation and Psychology Workshop*, Springer, pp. 273–282.

Kenneth Ward Church and Patrick Hanks (1990), Word Association Norms, Mutual Information, and Lexicography, *Computational Linguistics* **16** (1), pp. 22–29.

Kevin Lund and Curt Burgess (1996), Producing high-dimensional semantic spaces from lexical co-occurrence, *Behavior Research Methods, Instruments, and Computers* **28** (2), pp. 203–208.

Klaus Rothenhäusler and Hinrich Schütze (2007), Part of Speech Filtered Word Spaces, *in* Marco Baroni and Alessandro Lenci and Magnus Sahlgren, editor, *Proceedings of the 2007 Workshop on Contextual Information in Semantic Space Models: Beyond Words and Documents*, pp. 25–32.

Kris Heylen and Yves Peirsman and Dirk Geeraerts and Dirk Speelman (2008), Modelling Word Similarity: An Evaluation of Automatic Synonymy Extraction Algorithms, *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

Lillian Lee (1999), Measures of Distributional Similarity, *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 25–32.

Lonneke van der Plas and Gosse Bouma (2005a), Automatic Acquisition of Lexico-Semantic Knowledge for QA, *Proceedings of the IJCNLP workshop on Ontologies and Lexical Resources*, pp. 76–84.

Lonneke van der Plas and Gosse Bouma (2005b), Syntactic Contexts for Finding Semantically Related Words, *Proceedings of CLIN 04*, pp. 173–186.

Lonneke van der Plas and Jörg Tiedemann (2006), Finding Synonyms Using Automatic Word Alignment and Measures of Distributional Similarity, *Proceedings of the COLING/ACL*, pp. 866–873.

Magnus Sahlgren (2006), *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*, PhD thesis, Institutionen för lingvistik, Stockholms Universitet.

Padó, Sebastian and Lapata, Mirella (2007), Dependency-Based Construction of Semantic Space Models, *Computational Linguistics* **33** (2), pp. 161–199.

Roeland J.F. Ordelman (2002), Twente Nieuws Corpus (TwNC), *Technical report*, Parlevink Language Technology Group. University of Twente.

Thomas K. Landauer and Susan T. Dumais (1997), A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge, *Psychological Review* **104** (2), pp. 211–240.

Van de Cruys, Tim (2005), Semantic Clustering in Dutch: Automatically inducing semantic classes from large-scale corpora, *in* Khalil Simaan and Maarten de Rijke and Remko Scha and Rob van Son, editor, *Proceedings of CLIN 05*, pp. 17–32.

Van de Cruys, Tim (2006), The Application of Singular Value Decomposition to Dutch Noun-Adjective Matrices, *in* Piet Mertens and Cedrick Fairon and Anne Dister and Patrick Watrin, editor, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*.

Van de Cruys, Tim (2007), Exploring Three Way Contexts for Word Sense Discrimination, *in* Marco Baroni and Alessandro Lenci and Magnus Sahlgren, editor, *Proceedings of the 2007 Workshop on Contextual Information in Semantic Space Models: Beyond Words and Documents*, pp. 33–41.

Van de Cruys, Tim and Villada Moirón, Begoña (2007), Lexico-Semantic Multiword Expression Extraction, *in* Peter Dirix and Ineke Schuurman and Vin-

cent Vandeghinste and Frank Van Eynde, editor, *Proceedings of the 17th Meeting of Computational Linguistics in the Netherlands*, pp. 175–190.

Vossen, Piek, editor (1998), *EuroWordNet: a multilingual database with lexical semantic networks for European Languages*, Kluwer, Dordrecht.

Warren Weaver (1955), Translation, *in* W.N. Locke and D.A. Booth, editor, *Machine Translation of Languages*, MIT press, pp. 15–23.

Will Lowe and Scott McDonald (2000), The direct route: Mediated priming in semantic space, *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, pp. 675–680.

Yves Peirsman and Kris Heylen and Dirk Speelman (2007), Finding semantically related words in Dutch. Co-occurrences versus syntactic contexts., *in* Marco Baroni and Alessandro Lenci and Magnus Sahlgren, editor, *Proceedings of the 2007 Workshop on Contextual Information in Semantic Space Models: Beyond Words and Documents*, pp. 9–16.

Yves Peirsman and Kris Heylen and Dirk Speelman (2008), Putting things in order. First and second order contexts models for the calculation of semantic similarity, *Actes des 9ièmes Journées internationales d'Analyse statistique des Données Textuelles (JADT 2008)*, pp. 907–916.

Zelig Sabbettai Harris (1954), Distributional structure, *Word* **10** (21), pp. 146–162.

Zhibiao Wu and Martha Palmer (1994), Verb semantics and lexical selection, *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–138.

8

Improving the lexical coverage of English compound adjectives

Improving the lexical coverage of English compound adjectives in syntactic parsing

Nelleke Oostdijk
Department of Linguistics, Radboud University Nijmegen

Abstract

The present paper addresses the question how in syntactic parsing the coverage of words in previously unseen text may be improved. The adjectives in English are presented here as a case study. Working on the assumption that most new words that are introduced into the language are constructed on the basis of already existing words through the application of word-formation processes, we investigate the role that different word-formation processes play, more specifically in the formation of adjectives in English. An analysis of adjectives in the BNC shows that in the case of adjectives compounding is the word-formation process that is most productive. Moreover, compound adjectives are not formed by combining bases at will; rather, a limited set of fairly simple rules apply that restrict the co-occurrence of bases. This makes it feasible to develop an approach for handling compound adjectives which is rather effective, as is evident from the results from a first implementation where of a set of 30,561 compound adjectives derived from the BNC, 88.68% were correctly identified

Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands, pp. 117–130
Edited by: Suzan Verberne, Hans van Halteren, Peter-Arno Coppen.
 Contact: n.oostdijk@let.ru.nl

as such. Incorporation of the rules in the grammar underlying the Pelican parser accounts for a 7.65% increase in the parser's coverage of a subset of 10,123 sentences taken from the Leipzig corpus.

8.1 Introduction

In many computational linguistic applications involving tagging and parsing the lexicon is critically important to the success of the application. In order to obtain maximal coverage of a text, there is a need for full-coverage of the words that appear in it. For the lexicon this requirement poses a problem as it is impossible to have in advance a complete inventory of all the words that may be encountered in previously unseen text, even for a morphologically 'poor' language such as English. No matter how large a lexicon is, it will at best be near-complete. Therefore, many approaches have resorted to having the lexicon work in tandem with some heuristics that provide a fall-back option for cases where a specific lexical item is unknown in the sense that it has not been included in the lexicon. The heuristics capitalize on the generalization of observed commonalities in items, often morphological features (prefixes and suffixes), but also the surrounding context and spelling cues like capitalization.[1] From experiences in parsing, however, we know that there are still numerous words that are missed out on, while it is not just proper names that are missing: unknown words may occur in each of the open word classes In the present paper we investigate the nature of these words, restricting ourselves to adjectives in English, and how they are best dealt with.
The organization of the paper is as follows: In Section 8.2 we introduce the different word-formation processes and describe how these manifest themselves in data from the British National Corpus (BNC). Since compounding appears to be the most productive word formation process, in Section 8.3 we then focus on compound adjectives. An initial manual classification of a subset of the data suggests that these compounds can be described by means of syntactic rules. We describe briefly how this can actually be done and what results were obtained in the actual implementation. Section 8.4 concludes this paper.

8.2 Word-formation processes

The inventory of lexical items that make up the lexicon of a language is continuously being expanded through the introduction of new words. While occasionally words are introduced that are completely new in the sense that they have not been constructed on the basis of known words, more commonly words are introduced that are constructed on the basis of already existing words through the application of word-formation processes. In the formation of adjectives in English specifically the following processes are involved (cf. Quirk et al. 1985, p. 1520):[2]

[1]For research on unknown word guessing carried out in the area of part-of-speech tagging, see for example Nakagawa et al. (2001), Orphanos and Christodoulakis (1999), Thede (1998), Tseng et al. (n.d.) and Weischedel et al. (1993).
[2]In what follows the term 'base' is used to refer to the minimal free form of a lexical item. We shall consider as base adjectives all lexical items that are adjectives in their minimal free form. Thus *old*

a. prefixation: putting a prefix in front of the base sometimes with, but most usually without, a change of word class; e.g. *un-dead*, *non-empty*, *over-eager*

b. suffixation: putting a suffix after the base, sometimes without, but usually with, a change of word class; e.g. *adjustable*, *financial*, *successful*, *historical*

c. conversion: assigning the base to a different word class with no change of form; in the case of adjectives this process typically concerns the adjectival use of present and past participles; e.g. *crusading*, *pounding*, *slurred*, *validated*

d. compounding: adding one base to another; e.g. *old-age*, *cost-conscious*, *historically-eclectic*, *civil-political*.

In addition to these four processes adjectives can be formed with the help of combining forms (e.g. *hispano-*, *bio-*, *climato-*). As Quirk et al. observe such forms "have the semantic characteristics of the first constituent in a compound but they resemble prefixes in mostly (...) being obligatorily initial, in having little or no currency as separate words, and in not normally being the stressed part of a complex word" (1985, p. 1520).

Conversion, it appears, has been perceived as unproblematic and consequently has received very little attention from lexicographers and computational linguists as it is assumed that any participle can occur as adjective. This sharply contrasts with derivation (prefixation and suffixation) and compounding. These processes have been given ample attention with different degrees of success, where it is apparent that they cannot be handled in the same manner. Thus, as regards derivation, in dictionaries derivational affixes usually occur as separate entries, while in morphological analyzers or word form lexicons used for NLP, knowledge about derivational morphology is applied for the analysis or generation of word forms. Compounding does not lend itself for this kind of approach. In actual practice therefore, we find that dictionaries include only compounds with (presumably) high currency, while in NLP heuristics are used to capture what are assumed to be the more common instances relying on the presence of a word-final adjectival suffix and a hyphen signaling apparent compounding.[3] Meanwhile, compound adjectives have repeatedly been reported to be particularly high frequent in some text genres, especially in news reportage, advertising, and also in poetry (Meijs 1975, Salzman n.d., Jackson 2006), where they serve to condense information. In order to gain insight into how different word-formation processes con-

and *young* are considered base adjectives, while *economic* and *useful* are considered to be instances of adjectival suffixation.

[3]Compound adjectives receive quite some attention from usage guidebooks such as the *The American Heritage Book of English Usage*. Here, however, the focus of attention is mostly on whether or not a compound adjective should be hyphenated, and not so much on the composition of compound adjectives.

tribute to the formation of adjectives we decided to investigate the adjectives that occur in the British National Corpus (BNC).

8.2.1 Adjectives in the BNC

We extracted our initial data set from the BNC word frequency list as compiled by Killgariff. The set comprises all 107,657 types with which the POS tag 'aj0' (denoting the word class of adjective) has been associated.[4] Together these types account for 6,264,673 tokens in the corpus. The adjectives show a typical Zipfian distribution. The adjective type *other* which can be found at rank 1 by itself accounts for 2.07% of the tokens, while the 59,591 hapax types (making up 55.35% of the total number of types) together only account for 0.95% of the total number of tokens.

When we consider our data set, we find that the most frequent types of adjectives appear to fall into two groups: one comprising base adjectives (*other*, *new*, *good*, etc.) and the other comprising adjectives that have been arrived at through derivation, more in particular suffixation (*different*, *important*, *national*, etc.). It is only at rank 69 that the first instance of conversion (*following*) can be found, while compounding does not appear until rank 290 where we find *long-term* as the highest ranked compound adjective. These observations, together with the experience we had with unknown words in parsing previously unseen text which most of the time were found to be compounds, led us to hypothesize that there might be a correlation between the word-formation processes on the one hand and the type frequency on the other hand, such that adjective types with higher frequencies can be explained more often in terms of derivation with very little compounding, while with adjective types with lower frequencies more often compounding will be involved at the expense of derivation. Conversion is here hypothesized to be evenly distributed throughout and not to show any frequency effect.

8.2.2 Word-formation processes related to the type frequency distribution

In order to test the word-formation*type-frequency hypothesis we undertook an analysis of adjective types with frequency 1, 5, 10, 15, 20, and 25 respectively. We manually classified each of the adjective types according to the word-formation process involved. The results of the classification are presented in Table 8.1.[5] Compounding indeed appears to be the process that is most productive in the sense that it is responsible for the largest number of hapaxes and thus for the greatest increase in the number of previously unseen types.[6] However, unlike what we hypothesized it is not derivation that gives way to compounding, but conversion:

[4]Thus we excluded all other types where the POS tagger had assigned the tag aj0-av0, aj0-nn1, aj0-vvd, aj0-vvg, or aj0-vvn, indicating a high degree of uncertainty as to the appropriateness of associating the word with the class of adjectives (av0 = adverb, nn1 = common noun singular, vvd = past tense verb, vvg = present participle, vvn = past participle).

[5]The number of instances reported here are the number of instances that remain after we have discarded from our data all apparent 'rubbish' (tokenization errors, spelling errors, foreign language data, etc.)

[6]On productivity, see also Plag (2003).

Table 8.1: Proportion of types with frequency (F) 1, 5, 10, 15, 20 and 25 resp. explained through word-formation processes

Word formation process	Type F1 *N*=54,591	Type F5 *N*=2,324	Type F10 *N*=818	Type F15 *N*=429	Type F20 *N*=234	Type F25 *N*=193
Conversion	4.26	17.77	22.74	25.87	21.79	31.61
Derivation	28.61	30.25	34.72	33.33	36.32	33.68
Compounding	64.09	47.59	37.53	35.43	35.90	33.16
Combining	2.92	3.87	4.40	3.03	4.27	0.52
Base ADJ	0.12	0.52	0.61	2.33	1.71	1.04
total	100.00	100.00	100.00	100.00	100.00	100.00

while derivation is distributed rather evenly across different frequencies, conversion occurs increasingly less frequently among low-frequent types. Now one could speculate that the present limitation to only the set of items that have been tagged unambiguously as aj0 might have a serious impact on the relative frequency of conversion, as often the tags aj0-vvn and aj0-vvg are assigned to tokens which exhibit conversion. However, this does not appear to be the case: even if we include all instances of tokens tagged aj0-vvn or aj0-vvg and consider these as conversion, the picture essentially remains the same (cf. Table 8.2).

Table 8.2: Proportion of types with frequency (F) 1, 5, 10, 15, 20 and 25 resp. that can possibly be explained through conversion (tags aj0, aj0-vvn, aj0-vvg)

Word formation process	Type F1 *N*=60,198	Type F5 *N*=3,142	Type F10 *N*=1,171	Type F15 *N*=620	Type F20 *N*=349	Type F25 *N*=302
Conversion	9.31	26.03	30.15	30.81	32.95	36.09

In the next section we investigate compound adjective types more closely, as they play a key role in the appearance of previously unseen words.

8.3 Compound adjectives

In the definition provided by Quirk et al. (1985) compounds are formed by combining one base with another. This seems to suggest that bases can be combined freely, without being bound by any restrictions. In order to check whether this is indeed the case we investigated all compound hapaxes in our data. To this end we extracted from our initial data set the subset of compound adjectives with frequency 1. The set comprises 34,987 items, only 3,829 of which are complex (i.e. multi-word) compounds combining more than two words,[7] 31,158 are simple

[7]Examples of complex compound adjectives are *easy-to-grasp*, *fun-to-wear*, *red-and-white-striped*, *suddenly-made-redundant*, and *very-low-fat*.

compounds combining two words, nearly all of which are hyphenated.[8] In what follows we focus exclusively on the set of 30,561 hyphenated simple compounds.

8.3.1 Simple compounds

The simple compound adjectives in our data roughly fall into five main groups.[9] A brief characterization of each of these groups is given below.

Group 1 comprises compound adjectives that take an adjective base as head and some other word class as first part. The adjective base is either a base adjective or an adjective arrived at by way of derivation. The head typically combines with a noun, numeral or an adverb. Typical examples are *application-dependent*, *cabinet-wide*, *four-dimensional*, *climate-relevant*, *overly-sensitive*, and *pharmalogically-active*. Quite frequently time adverbs occur as in *once-blind*, *ever-reluctant*, *still-resident*, *then-arthritic*.
Group 2 is formed by compound adjectives that are formed by combining two adjective bases. Examples are *cognitive-affective*, *classical-scholarly*, *chemical-physical*, *electric-acoustic*, and *Egyptian-Syrian*.
Group 3 comprises compound adjectives that are headed by adjective bases that have been arrived at through conversion. One subgroup consists of items where the head combines with a noun, adjective or adverb. Examples are *panic-driven*, *bug-infested*, *fresh-caught*, *money-generating*, *posh-looking*, *duly-authorised*, *forever-changing*. Another subgroup is formed by compounds headed by a present or past participle where the head combines with a particle. Examples: *agreed-upon*, *signed-off*, *trimmed-down*, *turning-away*, and *coming-down*.
Group 4 is made up of derivational compounds. The head of the compound is always a noun which is combined with an adjective, a noun or a numeral. To the combination the adjectival suffix *-ed* is added, giving the resulting word its adjectival status. Examples are *sunken-cheeked*, *missing-toothed*, *bare-fisted*, *single-platformed*, *metal-cased*, *leopard-sized*, *4-cornered*, and *six-fingered*.
Group 5 comprises compounds that are considered to be adjectives but are more peripheral to the class of adjectives than items falling within any of the other groups above. They are headed by a noun which is preceded by an adjective or a numeral. Examples are *close-attack*, *big-league*, *four-sensor*, *sixteen-page*.[10]

As is apparent from this description, there are clear indications that in combining bases to form compound adjectives there are underlying syntactic rules and semantic restrictions to be observed. This suggests that it should be feasible to develop a set of rules that describe how single token lexical items that are listed the lexicon may be combined so as to form compound adjectives.

[8] Non-hyphenated compound adjectives are always written as single items: e.g. *timesharing*, *soft-hearted*, *windswept*.
[9] There are some minor types that we shall not discuss here in detail.
[10] In our data we also find a large number of instances where a noun base is combined with another noun base (e.g. *author-subscriber*, *rugby-soccer*). Whether to consider these modifying compound nouns as adjectives in the literature is subject of discussion (cf. Bauer (1983): 210; Meijs (1975): 194).

In the next sections we follow up on this idea as we describe how it was implemented in the context of the Pelican parser and lexicon.[11]

8.3.2 Compounding rules

The grammar underlying the Pelican parser is an attribute grammar that in terms of rewrite rules describes the syntactic structures that occur in English. The grammar operates in tandem with a lexicon in which in principle should account for all possible word forms in the language. The interface between the grammar and the lexicon constitutes of a set of defining rules in the grammar which specify the different word classes that are included in the lexicon. In the case of adjectives, several subclasses are distinguished: apart from the subclass of common adjectives, adjectives are distinguished that originate from a present participle (*-ingp*) or past participle (*-edp*) form of a verb, while also the adjectives *such*, *very* and *worth* are distinguished as separate subclasses on the grounds of their idiosyncratic syntactic behaviour.

Where the grammar stops at the point where the lexicon is called upon, a typical rule looks as follows:[12]

```
c ADJ ADJECTIVE (AJP_TYPE, GRADABILITY):
   n listed token ADJ (AJP_TYPE, GRADABILITY).
```

The rule states that with a lexical category or word class adjective information is associated which informs us about the type of adjective ('AJP_TYPE'), whether or not the adjective is gradable and if it is, whether the form of adjective is 'absolute', 'comparative' or 'superlative' ('GRADABILITY'). The definition of the adjective as a 'listed token' refers to the specification of the individual items that belong to this particular word class as it is given in the lexicon.

Lexical entries in the lexicon take the following form:

```
"gorgeous"  n listed token ADJ (ajp_type_attributive|
              ajp_type_predicative, gradability_absolute)

"singing"   n listed token ADJ ingp(ajp_type_attributive,
              gradability_absolute|gradability_non_gradable)
```

In order to account for compound adjectives, we extended our grammar with a set of rules that was based on the observations we had made in the analysis of the

[11] The Pelican parser is an English wide-coverage rule-based parser that is being developed at Nijmegen University. See also http://lands.let.ru.nl/projects/pelican

[12] The initial 'c' in the example rule indicates that it concerns a (lexical) category. The abbreviation and long name are used for the sake of convenience. While the long name enhances the readability of the grammar, the abbreviation is used when the eventual output is represented in the form of a tree. All nodes that are associated with rules that have been prefixed with the letter 'n' will not appear in the eventual output.

BNC data (see Section 8.3.1). We included rules of the form[13]

```
c ADJ ADJECTIVE (AJP_TYPE, GRADABILITY):
(1) n listed token N (N_TYPE, N_CLASS, NUMBER),
    "-\--",
    n listed token ADJ (AJP_TYPE, GRADABILITY);
(2) n listed token ADJ (AJP_TYPE, GRADABILITY),
    "-\--",
    n listed token ADJ (AJP_TYPE, GRADABILITY1).
(3) n listed token NUM (NUM_TYPE, NUMBER),
    "-\--",
    n listed token N (n_type_common, n_class_other,
                                          number_sing),
    "-ed";
(4) n listed token N (N_TYPE, N_CLASS, NUMBER),
    "-\--",
    n listed token LV (complementation_motr,
         finiteness_pastpart, MOOD, NUMBER1, PERSON);
(5) n listed token ADJ (AJP_TYPE1, GRADABILITY),
    "-\--",
    n listed token N (N_TYPE, N_CLASS, NUMBER).
```

The features associated with the various word classes were used to restrict the possible combinations, as for example in the case of compound adjectives formed on the basis of a past participle form of a verb preceded by an adjective, where we required that the lexical verb should be mono transitive ('complementation_motr', rule 4).

8.3.3 Applying the rules

In order to measure to what extent incorporation of the rules in the grammar contributes to an improvement of the parser's coverage, we carried out two evaluations: one in which we applied the adapted version of our parser to the set of 30,561 simple compound adjectives that we had derived from the BNC, the other in which we applied the parser to a subset of 10,123 sentences that were taken from the Leipzig Corpus. In both cases the lexicon remained unchanged. The results are presented below.

Results obtained on compound adjectives from the BNC

In the case of the compound adjectives from the BNC, the fact that all items in the data set were lower case presented a problem since in the approach taken by the Pelican parser and lexicon the distinction between upper and lower case is taken to be significant.[14] Prior to parsing, therefore, we manually restored upper case characters where necessary. The overall coverage we obtained was 88.68%, i.e.

[13]The abbreviations are as follows: ADJ adjective, LV lexical verb, N noun, and NUM numeral. Examples of the compound adjectives covered by each of the alternatives are *user-adjustable*, *cultural-political*, *four-stringed*, *mafia-controlled*, *direct-action*.

[14]The distinction between upper case and lower case is particularly relevant for distinguishing between proper nouns/names and common nouns (John vs john).

27,100 compound adjectives were accounted for by the rules we had developed (cf. Table 8.3). 17,707 of these were analyzed unambiguously, that is, in each case a single rule applied. In the 30.74% of the cases (9,399 items) more than one rule was applied, thus yielding ambiguity at sub-word level. For 3,461 items the set of rules fails and apparently needs to be extended.

Table 8.3: Coverage of compound adjectives

		# items	% of total
Success	single analysis	17,707	57.94
	multiple analyses	9,393	30.74
	subtotal	27,100	88.68
Failure	no analyses	3,461	11.32
	total	30,561	100.00

Discussion

In what follows we shall first discuss the set of items which were unambiguously identified as compound adjectives. Next we turn to the ambiguous cases. We conclude our discussion with an analysis of the failures.

Unambiguous cases

Among the cases that were unambiguously identified as compound adjectives, the distribution over the various groups we distinguished in Section 8.3.1 appears to be rather unbalanced. As Table 8.4 shows, group 3 compounds are the largest group by far, making up 46.46% of the total number of compound adjectives in our data set. When we look at this group in more detail, we find that past participle forms occur much more frequently than present participle forms, while compounds are much more frequently formed by combining a noun and a participle than by combining an adverb and a participle (for details see Table 8.5). Where with group 3 compounds the adverb follows the participle, it is always a particle (e.g. *about*, *away*, *back*, *by*, *down*, *in*, *on*, *off*, *through*): *dozing-off*, *gearing-up*, *hidden-away*, *nailed-down*, *backed-off*, etc. Adverbs that precede the participle are always general adverbs: *freely-existing*, *rapidly-expanding*, *publicly-approved*, *freshly-scrubbed*, *cylindrically-shaped*, etc.

Ambiguous cases

In 9,393 cases (30.74% of the total number of 30,561 items) multiple rules could be applied. As a result, ambiguity was generated at the sub-word level. In itself sub-word level ambiguity is not considered to be problematic as it does not appear in the eventual output.[15] However, ambiguity may point to unforeseen

[15]Recall that the objective was to improve on the lexical coverage of previously unseen words, in this

Table 8.4: Distribution of compound adjectives over different groups (cf. Section 8.3.1)

group	description	# items	% of total
1	headed by ADJ	1,985	11.21
2	ADJ-ADJ	298	1.68
3	headed by -ingp or edp	8,226	46.46
4	'derivational compounds'	842	4.76
5a	headed by N: ADJ-NUM-N	3,273	18.48
5b	headed by N: N-N	2,926	16.54
rest	minor types	157	0.89
total		***17,707***	***100.00***

Table 8.5: Sub-groups for group 3

combination	# items	% of total
N-LVingp	869	10.56
N-LVedp	5,118	62.22
ADV-LVingp	181	2.20
ADV-LVedp	1,737	21.52
LVingp-ADV	21	0.26
LVedp-ADV	300	3.65
total	***8,226***	***100.00***

interaction,between rules, in which case we might want to adapt our rules.
What we found was that, where there was ambiguity, in most cases there were two competing rules (cf. Table 8.6). One major source of ambiguity is of course the multiple word class membership of various tokens. For example, a great many words can be either an adjective or a noun (e.g. *private*, *light*, *public*, *current*), a noun or a present participle (e.g. *sinking*, *smelling*, *counting*, *dressing*), or an adverb or an adjective (e.g. *half*, *well*) Ambiguity that arises from the rules themselves is found in cases like *dramatic-coloured* where forms ending in *-ed* (like *coloured*) can be retrieved in full from the lexicon or—alternatively—are described as formed on the basis of a noun (*colour*) to which the adjectival suffix *-ed* has been added.

Pairs of rules that were most frequently in competition with each other are the following:
These results did not give rise to wanting to adapt the present set of rules.

Failures

case adjective compounds.

Table 8.6: Ambiguous cases

# parses	# items	% of total (N=30,561)
2	7,237	23.68
3	1,315	4.30
4	758	2.48
5	74	0.24
6	9	0.03
sub total	***9,393***	***30.74***

Table 8.7: Competing rules

rule 1 vs rule 2	example	# items	% of total
N-LVingp vs N-N	*news-reporting*	1,281	17.70
ADJ-N vs N-N	*cold-chain*	1,215	16.79
N-LVedp vs ADJ-N-ed	*explosive-tipped*	1,028	14.20
N-ADJ vs N-N	*policy-variant*	981	13.56
N-ADJ vs ADV-ADJ	*half-hysterical*	391	5.40
ADV-LVedp vs N-LVedp	*well-invested*	228	3.15
other		2113	29.20
	total	***7,237***	***100,00***

In those cases where the set of rules failed to identify the item as a compound adjective, failure could be attributed to either of two causes:

1. (at least one) part of the compound was not in the lexicon. This was the case for 1,460 items (4.78% of the 30,561 items).

2. while in principle both constituent parts of the compounds were accounted for in the lexicon, there was no rule describing the particular combinations. This held for 2,001 items (6.55% of the 30,561 items).

As regards tokens that were (as yet) missing from the lexicon, many of these tokens that were acronyms or chemical substances. Another class of tokens that was frequently missing is constituted by adjectives deriving from proper names, such as *Trotskyite* as encountered in *half-Trotskyite*, or *Wagnerian* as in *near-Wagnerian*). Other missing tokens did not point to any systematic omissions.

Inspection of the group of 2,001 items where the rules failed to identify the compound adjective as such led to the observation that some of the failures were due to a too rigid formulation of particular rules. Thus 109 items failed due to fact

that the rule describing compounds that consisted of a numeral followed by a common noun excluded the class of common nouns that denote some kind of measure. Examples are *12-ft*, *36-km*, *40-lb*, and *100-mph*. In a similar fashion, the restriction to have a noun followed by the adjectival suffix *-ed* preceded by a numeral failed to take into account instances where instead of a numeral the quantifier *many* occurs: *many-stranded*, *many-tiered*, *many-tentacled*. Other failed items suggested that additional rules might be formulated. These include the following:[16]

- a compound adjective may be formed on the basis of an adjective and a present participle verb. Especially the verbs sound and smell appear particularly productive here: together they account for 82 failures, including for example *metallic-sounding*, *soppy-sounding*, *unusual-sounding*, *Irish-sounding*, *authentic-sounding*, *corrupt-smelling*, *fishy-smelling*, *milky-smelling*.
- a compound adjective may be formed by combining a preposition and a singular common noun; for example *before-tax*, *between-species*, *up-river* (69 items of the 2,001)

While an analysis of the set of items that the set of rules failed to identify as compound adjectives suggests that some rules should be formulated less restrictive while also additional rules may be formulated, it is important to keep in mind that this should be done with care since there is the risk that this may have an undesired side-effect in generating (additional) ambiguity.

Results obtained on sentences from the Leipzig Corpus

We are currently in the process of analyzing the Leipzig Corpus, a collection of one million sentences originating from various newspapers (Associated Press, Wall Street Journal, Financial Times, OTS News Ticker). In order to see to what extent the compounding rules for adjectives contribute to the success of the parser, we investigated a subset of 10,123 sentences that have been parsed successfully. In 774 cases (7.65%) the correct parse is arrived at by means of one of the compounding rules. From this we may conclude that the compounding rules are quite effective: without these rules, the parser would have failed to produce the correct parse.[17]

8.4 Conclusion

Our analysis of adjectives as they occur in the BNC shows that in the case of adjectives, compounding is the word-formation process that is most productive. Moreover, we find that compounds are not formed by combining bases at will;

[16] It proved difficult to generalize over larger sets of failed items. Below, the two examples of rules that one might consider adding cover a fair number of instances. Any other rule would at best account for a handful of items.

[17] Note that the impact that the compounding rules have here may be biased by the particular genre (news reportage). Future research should give insight to what extent this is indeed the case.

rather, a limited set of fairly simple rules apply that restrict the co-occurrence of bases. The introduction of handcrafted rules that call upon information that is already present in the lexicon provides a means to maintain control and guard over the quality of the lexical information while a substantial improvement is obtained in the lexical coverage of compound adjectives.

References

Bauer, L. (1983), *English Word-formation*, Cambridge University Press, Cambridge.

Jackson, M. (2006), Compound Adjectives in Arden of Faversham, *Notes and Queries* **53** (1), pp. 51–55.

Meijs, W. (1975), *Compound Adjectives and the Ideal Speaker-Listener. A Study of Compounding in a Transformational-Generative Framework*, North-Holland, Amsterdam.

Nakagawa, T., T. Kudoh, and Y. Matsumoto (2001), Unknown Word Guessing and Part-ofSpeech Tagging Using Support Vector Machines, *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS2001)*, Tokyo, pp. 325–331.

Orphanos, G. and D. Christodoulakis (1999), POS Disambiguation and Unknown Word Guessing with Decision Trees, *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL–99)*, pp. 134–141.

Plag, I. (2003), *Word-formation in English*, Cambridge University Press, Cambridge.

Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik (1985), *A Comprehensive Grammar of the English Language*, Longman, London.

Salzman, A. (n.d.), Using Compound Adjectives to Give Physical or Metaphorical Descriptions. Available at `http://www.iei.uiuc.edu/structure/Structure1/haired.html`.

Thede, S. (1998), Predicting Part-of-Speech Information about Unknown Words using Statistical Methods, *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (ACL/COLING–98)*, pp. 1505–1507.

Tseng, H., D. Jurafsky, and C. Manning (n.d.), Morphological features help POS tagging of unknown words across language varieties. Available at `http://www.stanford.edu/~jurafsky/sighan_pos.pdf`.

Weischedel, R., M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci (1993), Coping with Ambiguity and Unknown Words through Morphological Models, *Computational Linguistics* **19** (2), pp. 359–382.

Resources consulted

BNC database and word frequency lists (n.d.). Compiled by Adam Kilgariff. Available at `http://www.kilgarriff.co.uk/bnc-readme.html`.

Oxford English Dictionary (2003), Oxford University Press, Oxford.

The American Heritage Book of English Usage. A Practical Guide to Contemporary English (1996). Available at `http://www.bartleby.com/64/84.htm`.