Proceedings of the
Australasian Language Technology Summer School (ALTSS) and Australasian Language Technology Workshop (ALTW) 2003

University of Melbourne, Australia - December 8-12, 2003

Editors: Catherine Bow and Baden Hughes

Australasian Language Technology Association

ISBN: 0-9751687-1-1

Preface

The Australasian Language Technology Association is proud to present its inaugural Summer School and Workshop. The Summer School consists of eight intensive courses and eight standalone lectures presented by experts in the field, and targetted at postgraduate students and researchers from academia and industry. The Workshop, chaired by Alistair Knott and Dominique Estival, provides a forum for the presentation and discussion of new research in language technology. On Wednesday evening, the Language Technology Forum will promote language technology to the wider community, as a field which is addressing fundamental questions in cognitive science and generating important new social applications.

At a time when the field of language technology is experiencing unprecedented growth in Asia, Europe and North America, it is encouraging to see a healthy community taking shape in Australasia. The isolation of our region, its linguistic diversity, and its rapid uptake of new technologies, present important challenges and opportunities. With timely cooperation in research and training, such as the events on offer this week, the language technology community will continue to expand. Major goals will be to develop more natural human-machine interfaces, and more efficient ways to access the information contained in large collections of text and speech. Progress in these areas will lay the groundwork for new applications which address the challenges and opportunities of our region and, more generally, support the multilingual information society of the future.

On behalf of the ALTA Executive Committee, I would like to thank all the speakers, sponsors, editors, and participants for making this week a success.

Associate Professor Steven Bird
Department of Computer Science and Software Engineering, University of Melbourne
President, Australasian Language Technology Association

- Index
- Table of Contents
- Acknowledgements

Acknowledgements

We would like to acknowledge the following people and organisations without which this event would not have been possible:

- the presenters of courses and lectures, all of whom volunteered their time;
- the ALTA Executive for their support and advice on the organisation of the event;
- Alistair Knott and Dominique Estival for their organisation of Australasian Language Technology Workshop;
- Bronwen Hewitt and Jen Westphal at the University of Melbourne Conference Management for logistical support;
- the Department of Computer Science and Software Engineering at the University of Melbourne for administrative support;
- Professor Rao Kotagiri for his personal endorsement and support of opportunities for postgraduate research training through events such as this;
- the University of Melbourne for hosting the event;
- Defence Science and Technology Organisation (DSTO) for their generous sponsorship of the Australasian Language Technology Workshop.
- Appen Pty Ltd for their generous sponsorship of the Australasian Language Technology Summer School.

- Index
- Preface
- Table of Contents

Proceedings of the Australasian Language Technology Workshop

Volume 1, Melbourne, December 2003
Online Edition
ISSN 1834-7037

Workshop Papers

M Herke-Couchman and C Whitelaw (Sydney)
Identifying interpersonal distance using systemic features

Incremental Chart Parsing with Predictive Hints

Rolf Schwitter
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
schwitt@ics.mq.edu.au

Abstract

This paper describes an incremental chart parser that generates look-ahead categories on the fly for a controlled natural language. These predictive hints tell the author what kind of syntactic (or semantic) structure can follow the current input string and thereby aim at helping the author to reduce the cognitive burden to learn and remember the rules of the controlled language. The parser can handle modifications (insertion, deletion, and replacement) to the input string without the need to reparse the entire string. These modifications are a function of the size of the tokens changed rather than the size of the entire input.

1 Introduction

Over the last decade two types of controlled natural languages have been developed: human-oriented controlled languages and machine-oriented controlled languages (Huijsen, 1998; O'Brien, 2003).

The main goal of human-oriented controlled natural languages is to improve the readability and understandability of technical documents for human readers, especially non-native speakers (AECMA, 2001). Machine-oriented controlled natural languages, on the other hand, try to ease the translation process (Kamprath, 1998) and to facilitate the subsequent inference processes (Fuchs and Schwertel, 2003; Schwitter et al., 2003; Sukkarie, 2003).

In general, a controlled natural language can be defined as a subset of a natural language that has been restricted with respect to its grammar and its lexicon. Grammatical restrictions usually result in less complex and less ambiguous texts. Lexical restrictions reduce the size of the vocabulary and the meaning of the lexical entries for a particular application domain.

To allow writing unambiguous and precise specifications, we have developed a machine-oriented controlled natural language (PENG – Processable ENGlish) for knowledge representation (Schwitter, 2002). Specifications written in PENG can be translated unambiguously into first-order predicate logic via discourse representation structures and can be automatically checked for consistency and informativeness with the help of third-party reasoning services (McCune, 2001; Bos, 2001; McCune, 2003; Bos, 2003).

It is well known that writing documents in a controlled natural language is hard and time-consuming without the support of intelligent writing assistance (Goyvaerts, 1996; Power et al., 2003). To ease the writing process and to guarantee well-formed syntactic structures, PENG uses a look-ahead editor that displays after each word form a set of syntactic categories that inform the author how the input string can be continued (Schwitter et al., 2003).

The look-ahead editor of PENG communicates with a chart parser that processes a unification-based grammar. The chart parser generates these predictive hints dynamically while the text is written and thereby enforces the restrictions placed upon the language. Apart from these hints, the chart parser also generates for each input string a discourse representation structure (Kamp and Reyle, 1993) as well as a paraphrase (Schwitter and Ljungberg, 2002) that reflects the interpretation of the machine in controlled natural language.

So far, it has not been possible to modify a substring of the text without having to reparse the entire text and regenerate the output (discourse representation structure and paraphrase) from scratch. This is unsatisfactory and demands for a more sophisticated approach to deal with modifications. It would be desirable to be able to edit the text during the writing process without the need for extensive reparsing. The chart parser should be able to handle modifications (insertion, deletion, and replacement) as efficiently as possible and to do its job in a piecemeal fashion constructing the representation of the text word by word and providing, at the same time, look-ahead categories that are contingent on the current input.

We will call such a chart parser that is capable of constructing a representation bit by bit and handling modifications without the need for exhaustive reparsing and reconstructing of the underlying representation an *incremental chart parser*.

Ideally, the time that the incremental parser spends to process a modification of arbitrary length (without violating the approved rules of the controlled language) should be proportional to the complexity of the change.

The reminder of this paper is organised in the following way: In Section 2, I will set up the requirements for an incremental chart parser. In Section 3, I will discuss the properties and shortcomings of a naïve chart parser for the task at hand. In Section 4, I will present the benefits of an incremental chart parser and show that the intended solution fulfills the requirements that have been specified in Section 2. In Section 5, I will present update handling algorithms for generating look-ahead categories and for dealing with modifications. In Section 6, I will evaluate the introduced algorithms and compare them with naïve reparsing. Finally, in Section 7, I will summarize the advantages of the presented approach and give some indicators for further research.

2 Requirements to an Incremental Chart Parser for PENG

PENG is a machine-oriented controlled natural language designed to write precise specifications and aims at supporting the knowledge acquisition process for various tasks (Schwitter, 2002). PENG consists of a strict subset of standard English. The restrictions of the language are defined with the help of a controlled grammar and a controlled lexicon, and enforced by a look-ahead editor (Schwitter et al., 2003).

As shown in the dataflow diagram for the PENG system in Figure 1, sentences written in PENG are first sent to the controlled language (CL) processor and then translated into first-order predicate logic via discourse representation structures. This logical representation is subsequently checked for consistency and informativeness with the help of a theorem prover and a model builder.

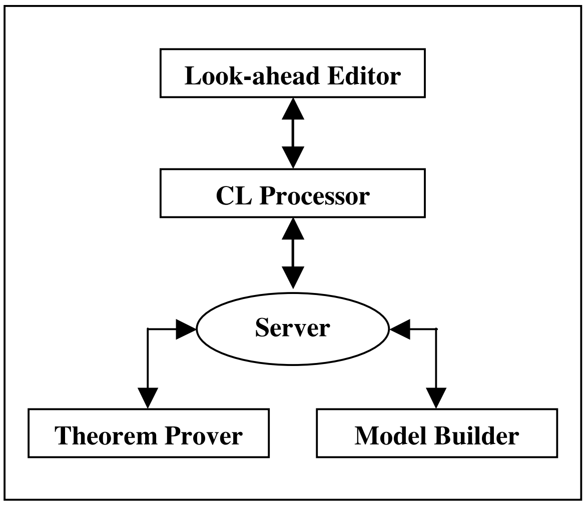


Figure 1: Dataflow diagram for the PENG system

The look-ahead editor of PENG communicates with the CL processor (chart parser and unification-based grammar) via a socket interface. The CL processor is running as a client process and is connected via a server with a theorem prover (OTTER; McCune, 2003) and a model builder (MACE; McCune 2001). The theorem prover and the model builder are both running separate client processes.

One of the deciding factors for the acceptability of a controlled language is the availability of automatic writing assistance. Texts in controlled natural language should not only be easy to write but also easy to modify without the need for time-consuming reprocessing.

This is why we need a processing strategy that informs the author about the permissible structure of the text and that supports basic editing operations in an efficient way.

2.1 Look-ahead categories

Given a set of grammar rules that define a controlled natural language such as PENG, the incremental chart parser should be able to generate a set of look-ahead categories after each word form that the author enters. For example, for the sentence

1. *The customer inserts a credit card.*

the following look-ahead categories should be generated (simplified here):

The [adjective | noun]

… *customer* [relpron | negation | verb]

… *inserts* [determiner]

… *the* [adjective | noun]

… *credit* ['card' | relpron | prep | conj | '. ']

… *card* [relpron | prep | conj | '. ']

In this example, all the look-ahead categories are either lexical categories or word forms. However, the algorithm should be easily parameterizable so that predictive categories for various syntactic strata as well as for semantic information can be generated.

2.2 Insertion

The insertion operation should allow the addition of an arbitrary number of words into the input string as long as this modification does not violate the rules of the controlled language. For example, the insertion of the relative pronoun *who* into sentence *1* should lead to the noun phrase:

2. *The customer who inserts the credit card …*

that is part of the controlled natural language PENG.

Note that the insertion results here in a categorial change; instead of a complete sentence we have now to deal with a complex noun phrase. This means that the punctuation mark introduced by the sentence needs to be removed automatically and a new set of look-ahead categories has to be generated and displayed for the last word form of the input string:

… *card* [relpron | prep | verb | conj]

As this example shows, the noun phrase in *2* could now be continued with a relative clause (*3*), a prepositional phrase (*4*), a verb phrase (*5*), or a coordinated nominal structure (*6*):

3. *… that is valid owns a password.*
4. *… into the slot owns a password.*
5. *… owns a password.*
6. *… and enters the PIN owns a password.*

Each of these sentences constitutes a well-formed structure in the controlled natural language.

2.3 Deletion

The deletion operation should allow the cutting of an arbitrary sequence of words in the input string. For example, the deletion of the prepositional phrase *into the slot* in

6. *The customer who inserts the credit card into the slot owns a password.*

should result in a well-formed sentence in controlled language:

7. *The customer who inserts the credit card owns a password.*

The author should be able to delete a substring first by highlighting it and then by cutting it:

8. *The customer who inserts the credit card into the slot owns a password.*

This design decision reduces the complexity of the algorithm, since it results in one single cutting event that tells the parser when recomputation should be resumed.

2.4 Replacement

If the insertion and deletion operation are in place, then the infrastructure for the replacement operation exists. In essence, the replacement operation is a deletion followed by an insertion operation. For example, the replacement of the compound noun *credit card* with *MasterCard* in sentence *1* results in

9. *The customer inserts the MasterCard.*

It seems that a replacement operation is more complex than a deletion or an insertion operation, since these two operations need to be applied in sequence. However, as we will discuss in detail in Section 5, this is not the case, since the replacement operation can be implemented in a way that does not demand for extensive reprocessing.

3 Chart Parsing

Parsing is the process of analyzing the syntactic structure of an input string and has traditionally been understood as a batch-mode process.

The problem with any naïve parsing algorithm – independent of the parsing strategy – is the unnecessary repetition of work that will occur for processing any non-trivial grammar.

Suppose a top-down parser is attempting to parse the sentence:

10. The password is valid.

Given the following simple (context-free) phrase structure grammar

```
s  → np, vp.
np → det, noun, rc.
np → det, noun.
rc → relpro, vp.
vp → verb, adj.
```

the parser will first attempt using the rule

```
np → det, noun, rc.
```

and then after failing with that rule, it will try the alternative rule

```
np → det, noun.
```

That means the parser will repeat the work of analyzing the determiner (`det`) and the noun (`noun`) once for each rule.

An active chart parser (Kay, 1980; Gazdar and Mellish, 1989; Ferro and Pardo, 1995) avoids this repetition of work by storing information about well-formed substrings as well as information about substring hypotheses that it has partially explored in a table (chart). The chart parser can then look up these substrings in the chart and expand them – if necessary – instead of recomputing them.

Given an input string *I* and a grammar *G*, we can define a chart as a set of edges where an edge is a triple of the form $<v_i,\ v_t,\ R>$. The first two elements v_i and v_t are integers and represent starting and ending vertices of *I* or of a substring of *I* and *R* represents a dotted rule. A dotted rule is a rule of the form $X \rightarrow \alpha \bullet \beta$ and corresponds to an *X* edge containing an analysis of confirmed constituents α that are seeking for constituents β.

For example, if `s → np, vp.` is a rule of the grammar and sentence *10* is an input string, then the first two dotted rules below represent unconfirmed hypotheses while the third rule represents a fully confirmed hypothesis:

```
<0,0,s → ● np vp>
<0,2,s → np ● vp>
<0,4,s → np vp ●>
```

Edges that correspond to unconfirmed hypotheses are known as *active edges* and those that correspond to confirmed hypotheses as *inactive edges*.

The basic operation of a chart parser involves combining an active edge with a completed inactive edge. The result is either a new inactive edge or a new active edge that spans both the active and inactive edges. This fundamental rule cannot be applied to a chart that contains no edges. Before anything can happen, an initialization process needs to set the chart up with inactive word edges and a rule invocation strategy needs to be defined that creates new active edges as a result of the application of the fundamental rule.

A chart parser usually uses an agenda to keep track of the edges that need to be processed. Such an agenda can be thought of as a list of edges. Adding new edges to the front of the agenda leads to a depth-first search strategy and adding them to the end would lead to breath-first search.

In our implementation, edges are stored in the following modified format:

edge(ID,v_s,v_t,LHS,RHSL)

ID is an integer that stands for a sentence identifier. *LHS* represents the category on the left hand side of a dotted rule and *RHSL* represents a list of unconfirmed daughter categories on the right hand side of the rule. If *RHSL* is empty (*[ ]*), then the edge is inactive, otherwise active.

For example, a top-down chart parser will produce the following edges[1] for sentence *10* using our simplified grammar rules introduced above:

```
edge(0,0,s,[np,vp])
edge(0,0,np,[det,noun])
edge(0,1,np,[noun])
```

[1] The sentence identifier is not displayed in the edges. The categories of the grammar are atomic and do not contain any additional syntactic or semantic arguments. The grammar is not complete, since preterminal rules such as `det → [the]` are missing. As a consequence inactive edges such as `edge(0,1,det,[])` do not appear in the simplified chart.

```
edge(0,2,np,[])
edge(0,2,s,[vp])
edge(2,2,vp,[verb,adj])
edge(2,3,vp,[adj])
edge(2,4,vp,[])
edge(0,4,s,[])
edge(0,0,np,[det,noun,rc])
edge(0,1,np,[noun,rc])
edge(0,2,np,[rc])
edge(2,2,rc,[relpro,vp])
```

Although such a batch-mode chart parser avoids repeating work and keeps active and passive edges in the chart, it cannot deal with modifications to the current input string without reprocessing the entire string.

4 Incremental Chart Parsing

An incremental chart parser, by contrast, can handle modifications to an input string that it has already parsed without having to reprocess the entire string from scratch. The key idea of incremental chart parsing is to use information about edge dependencies for keeping track of edges that have to be updated (Wirén, 1989; Wirén 1994).

Let us explore this idea by an example and then refine it. Suppose we modify sentence *10* by inserting the relative pronoun *that* between the noun phrase and the verb phrase, then we get a complex noun phrase as result:

11. The password that is valid …

In comparison to the chart for sentence *10* in Section 3, the processing of this noun phrase results in 4 new edges

```
edge(2,3,rc,[vp])
edge(2,5,rc,[])
edge(0,5,np,[])
edge(5,5,vp,[verb,adj])
```

and in 4 modified edges (with modifications in bold face)

edge(**3**,**3**,vp,[verb,adj])
edge(**3**,**4**,vp,[adj])
edge(**3**,**5**,vp,[])
edge(0,**5**,s,[**vp**])

We can make the following observations when we compare the charts for sentence *10* and for the noun phrase *11* in more detail:

- The active edge `edge(0,2,np,[rc])` for sentence *10* hypothesizing that *the password* was the beginning of a noun phrase followed by a relative clause has been expanded to an inactive edge `edge(0,5,np,[])` to cover the relative clause in *11*.
- All the edges that make up the noun phrase *the password* in sentence *10* remain unaffected by the modification.
- All the edges that make up the verb phrase *is valid* in sentence *10* remain unaffected apart from the indices of the vertices (displayed in bold face) that have been updated.
- The passive edge `edge(0,4,s,[])` representing sentence *10* has been replaced by an active edge edge(0,**5**,s,[**vp**]), because *11* is a noun phrase and not a complete sentence.

In summary, we can state that there is no need to recompute an edge, if that edge does not in any way depend upon the vertices that have been changed or on any edges that were based on those edges.

A closer look into the chart for sentence *10* reveals that the edge `edge(0,4,s,[])` is the only one that spans the vertex (insertion point) where the relative pronoun would be inserted. This suggests the following informal solution to process the modification:

1) Find all edges on the right hand side of the insertion point, in our case all those edges whose starting vertex is greater than or equal to the insertion point, and create a new subchart *CR* for them.
2) Renumber all starting and ending vertices of the edges in *CR* to be v_{s+1} and v_{t+1}.
3) Find all edges on the left hand side of the insertion point, in our case all those edges whose ending vertex is smaller than or equal to the insertion point, and create a new subchart *CL* for them.
4) Create a new chart *C* by appending the subchart *CR* to the end of the subchart *CL*.
5) Create new hypotheses beginning at the insertion point for the word form *that*.

6) Reparse the string, using only the new edges in the agenda and the new chart *C*.

Note that this solution automatically excludes edges such as `edge(0,4,s,[])` from the new chart *C*, since we considered only edges that do not bridge the insertion point. At first glance, it seems that an optimization should be possible, since not all edges in the subcharts are affected by the editing operation. For example, only the modified edge `edge(3,5,vp,[])` in the subchart *CR* spanning the verb phrase on the right hand side of the insertion point takes part in reparsing. Similar observations can be made for the subchart *CL* where only those edges that end at the insertion point are affected by reparsing. However, it turns out that first filtering the subchart *CR* and then reconstructing the entire chart after parsing is costly and does not result in a speed-up of parsing in comparison with the unfiltered version of the chart.

5 Update Handling Algorithms

After this informal discussion of the problem, I will give a more formal description of the update handling algorithms for finding look-ahead categories and for dealing with modification (insertion, deletion, and replacement).

5.1 Finding look-ahead categories

Look-ahead categories are generated after each word form that the author enters or whenever an approved modification results in a syntactic structure that needs to be completed by the author.

Formally, a set of look-ahead categories *LC* for a word w ending at vertex v_i can be calculated in the following way:

1) Find all active edges ending at v_i.
2) For each active edge:
   a) Select the *RHSL* of remaining categories.
   b) For the first category in *RHSL*, check if it is a lexical category:
      i) If yes, then store the solution in *LC*.
      ii) If not, find a rule that rewrites the category into further categories, then select the first category and return to 2b.

Apart from lexical categories, it is also possible to collect other categories, for example non-terminal categories by extracting them from the grammar rules, or semantic categories if they are stored in the lexicon and accessible via lexical categories. Collecting also look-ahead categories for non-terminal symbols in Step 2ii results in a list of hierarchically order categories and eases customization of this functionality for the user interface.

5.2 Editing operations

According to our definition, the incremental chart parser should not only be able to handle piecemeal additions to a string but also to handle arbitrary modifications efficiently. Ideally, the time that the incremental algorithm uses for processing a modification should be a function of the size of the modification rather then the size of the entire input. In simple words: a small modification should require less work than a big modification. Note that the algorithms presented below for the editing operations do not explicitly delete bridging edges but rather exclude them by reconstructing the chart.

Insertion

Inserting a word w at a vertex v_i in a string can be calculated in the following way:

1) Find all edges for which the index of the starting vertex v_s is greater than or equal to v_i, that is $s \geq i$, and create a new subchart *CR* for them.
2) For all the edges in *CR*
   a) renumber the starting vertex to be v_{s+1},
   b) renumber the ending vertex to be v_{t+1}.
3) Find all edges for which the index of the ending vertex v_t is smaller than or equal to v_i and the starting vertex v_s is not equal to v_i, that is $t \leq i \wedge s \neq i$, and create a new subchart *CL*.
4) Create a new chart *C* by appending the subchart *CR* to the end of the subchart *CL*.
5) Create new hypotheses beginning at v_i for each category that the new word w belongs to.
6) Reparse the string, using only these new edges as the agenda for the parser, and providing it with the updated chart *C*.

Deletion

Deleting a word w at a vertex v_i in a string is to some extent similar to the reverse of inserting a word. The algorithm looks as follows:

1) Find all edges for which the index of the starting vertex v_s is greater than or equal to v_i, that is $s \geq i$, and create a new subchart *CR* for them.
2) For all the edges in *CR*
    a) renumber the starting vertex to be v_{s-1},
    b) renumber the ending vertex to be v_{t-1}.
3) Find all edges for which the index of the ending vertex v_t is smaller than v_i and the starting vertex v_s is not equal to v_i, that is $t < i \wedge s \neq i$, and create a new subchart *CL1*.
4) Find all edges for which the index of the ending vertex v_t is equal to v_i, that is $t = i$, and create a new subchart *CL2*.
5) Create a new chart *C* by appending the subchart *CR* to the end of the subchart *CL1*.
6) Reparse the string, using the subchart *CL2* as the agenda for the parser, and providing it with the subchart *C* as new chart.

Note that the agenda above consists of the subchart *CL2* in contrast to the insertion operation where the agenda for reparsing consists only of the new word hypothesis.

Replacement

Replacing a word *w* at a vertex v_i in a string can be described as a deletion followed by an insertion operation. This is what authors do when they replace a word in an interactive text editor. They first delete the word and then insert a new word.

However, simply executing these two operations in sequence would not be very efficient. For example, if the word *customer* is replaced with *client* in sentence *10*, then all words lie between the same vertices as they did before the replacement operation. In this case, the chart does not need to be partially recreated twice, since the first recreation will renumber vertices and create edges that will immediately be reset or deleted again.

This observation results in the following optimized algorithm:

1) Create new hypotheses beginning at v_i for each category *Cat* that the new word *w* belongs to.
2) Replace the inactive word edge *E* in the chart starting at vertex v_i with the new word edge *E'* so that the categories of *E* and *E'* are identical.

This is a significant improvement over executing the two operations in sequence.

6 Evaluation

The presented algorithms that incrementally update the chart result in a speed-up for all modification operations compared with naïve reparsing of the input string after an editing operation. The average improvement for insertion is of a factor of 1.44, for deletion 1.28, and for replacement 17.64. As the results show, replacement can be implemented very efficiently. As already mentioned, additional filtering of the subchart *CR* does not result in any speed-up. It seems that in our Prolog implementation renumbering of the vertices in the affected edges is the biggest cost factor, since this involves arithmetic operations and not pure unification.

7 Conclusion

In this paper, I discussed an incremental chart parser that generates predictive hints and allows for arbitrary editing operation as long as the result is an approved structure in controlled natural language. The generated look-ahead categories consist of syntactic (or semantic) categories and aim at supporting the writing process of the controlled natural language. These predictive hints ensure that the author follows the rules of the controlled natural language and guarantee unambiguous and precise texts (in our case "seemingly informal" specifications).

The editing operations (insertion, deletion, replacement) are bound to the affected part of the string and require only minimal reparsing. This means that the modifications are a function of the size of the words changed rather than the size of the entire text. The current solution deals only with local updates. In the future, I would like to look into the problem of updating anaphoric references in the text and in the underlying discourse representation structure after a nominal expression has been modified. The goal is to find a solution that does not require extensive reparsing of the input text.

Acknowledgements

I would like to thank the students (Jason Barles, James Cameron, Dennis Chan, Mary Gardiner, Hogan Ho, and Simon Kissane) who were enrolled in the honours class *COMP448: Advanced Topics in Natural Language Processing* in the summer semester 2003 at Macquarie University for their enthusiasm in exploring various directions in incremental chart parsing. I also thank two anonymous reviewers of this paper for their constructive comments.

References

AECMA. 2001. The European Association of Aerospace Industries. AECMA Simplified English, AECMA Document PSC-85-16598. A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language. Issue 1, Revision 2, 15 January.

J. Bos. 2001. DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures, in: Blackburn and Kohlhase (eds.): IcoS-3. Inference in Computational Semantics. Workshop Proceedings, Siena, Italy, June.

J. Bos. 2003. Exploring Model Building for Natural Language Understanding. Proceedings of the Fourth International Workshop on Inference in Computational Semantics (ICoS-4), September 25-26, INRIA Lorraine, Nancy.

M. V. Ferro, M. A. A. Pardo. 1995. Exploring interactive chart parsing. Procesamiento del Lenguaje Natural, Bilbalo, Spain, pp. 158-172.

N. E. Fuchs, U. Schwertel. 2003. Reasoning in Attempto Controlled English. Proceedings of the Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003), Mumbai, India.

G. Gazdar, C. Mellish. 1989. Natural Language Processing in Prolog. An Introduction to Computational Linguistics. Addison-Wesley, Wokingham, England.

P. Goyvaerts. 1996. Controlled English, Curse or Blessing? – A User's Perspective. Proceedings of the First International Workshop on Controlled Language Applications, March 26-27, Leuven.

W. O. Huijsen. 1998. Controlled Language – An Introduction. Proceedings of CLAW 1998, Pittsburgh, pp. 1-15.

H. Kamp, U. Reyle. 1993. From Discourse to Logic. Kluwer, Dordrecht.

C. Kamprath, T. Mitamura, E. Nyberg. 1998. Controlled Language for Multilingual Document Production: Experience with Caterpillar Technical English, Proceedings of the Second International Workshop on Controlled Language Applications, Pittsburgh.

M. Kay 1980. Algorithm Schemata and Data Structures in Syntactic Processing. Report CSL-80-12, Xerox Parc, Palo Alto, California.

W. McCune. 2001. MACE 2.0 Reference Manual and Guide. ANL/MCS-TM-249. Mathematics and Computer Science Division, Technical Memorandum No. 249, Argonne National Laboratory, Argonne.

W. McCune. 2003. OTTER 3.3 Reference Manual. ANL/MCS-TM-263, Mathematics and Computer Science Division, Technical Memorandum No. 263, Argonne National Laboratory, Argonne.

S. O'Brien. 2003. Controlling Controlled English, An Analysis of Several Controlled Language Rule Sets. Proceedings of EAMT-CLAW03, Controlled Language Translation, May 15-17, Dublin City University, pp. 105-114.

R. Power, D. Scott, A. Hartley. Multilingual Generation of Controlled Languages. Proceedings of EAMT-CLAW03, Controlled Language Translation, May 15-17, Dublin City University, pp. 115-123.

R. Schwitter. 2002. English as a Formal Specification Language. Proceedings of the Thirteenth International Workshop on Database and Expert System Applications (DEXA 2002), Aix-en-Provence, France, pp. 228-232.

R. Schwitter, A. Ljungberg. 2002. How to write a document in controlled natural language. Proceedings of the Seventh Australasian Document Computing Symposium, Sydney, Australia, December, pp. 133-136.

R. Schwitter, A. Ljungberg, D. Hood. 2003. ECOLE: A Look-ahead Editor for a Controlled Language. Proceedings of EAMT-CLAW03, Controlled Language Translation, May 15-17, Dublin City University, pp. 141-150.

J. Z. Sukkarieh. 2003. Mind your Language! Controlled Language for Inference Purposes. Proceedings of EAMT-CLAW03, Controlled Language Translation, May 15-17, Dublin City University, pp. 160-169.

M. Wirén. 1989. Interactive incremental chart parsing. Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics, Manchester, England, pp. 241-248.

M. Wirén. 1994. Bounded incremental chart parsing. Clause Report 36. Universität des Saarlandes, Saarbrücken, Germany.

One-Anaphora and the Case for Discourse-Driven Referring Expression Generation

Robert Dale
Centre for Language Technology
Macquarie University
Sydney NSW 2109
`Robert.Dale@mq.edu.au`

Abstract

Conventional approaches to the generation of referring expressions place the task within a pipelined architecture, typically somewhere between text planning and linguistic realisation. In this paper, we look at the issues that arise in generating *one*-anaphoric referring expressions; examination of this task causes us to reflect on the current predominant architectural models for natural language generation, and leads us to suggest an alternative architecture where decisions that influence forms of reference happen much earlier in the process of natural language generation.

1 Introduction

Referring expression generation is a much-explored task within natural language generation: given an internal symbol that corresponds to an entity in some real or imagined world, we need to work out what properties of that entity should be used to describe the entity so that our hearer will be able to identify it as the intended referent. Many different algorithms have been developed to address this task, which is generally conceived of as mapping from a symbol—effectively, a referent—to a set of properties—a sense. The computation of the appropriate set of properties to use takes account of the other potential referents in the context, selecting properties which rule these distractors out of consideration.

Conventional approaches to the process of generating referring expressions place the task within a pipelined architecture, where it is assumed that questions of what content should be conveyed in a text are resolved before questions of surface form are considered; this is the well-known STRATEGY VS TACTICS distinction first discussed in the context of natural language generation in the mid-1970s. However, it is not clear exactly where in the pipeline the process of generating referring expressions should belong. In Reiter and Dale [2002], reflecting current practice in the field, we positioned it in the microplanning stage, where microplanning is an intermediate stage lying between text planning and surface realisation. Even there, however, we noted that there are interactions between the three microplanning tasks of sentence planning, lexical selection and referring expression generation that argue for a more interleaved constraint-based approach to the problem.

The principal focus of existing work has been the generation of definite noun phrase references; relatively little has been written on generating other kinds of referring expressions. In particular, there is virtually no work on the generation of *one*-anaphora. Taking up some ideas first explored in [Dale 1992, 1995], this paper looks at how a consideration of where *one*-anaphora fits into the generation process might cause us to review the kinds of architecture that are required for natural language generation.

Section 2 first summarises the conventional approach to referring expression generation, and reviews how this fits into the standard architectural models for natural language generation. Section 3 introduces the phenomenon of *one*-anaphora, before going on to explore how the generation of *one*-anaphora might be integrated into existing approaches of referring expression generation. Section 4 then suggests an alternative approach, where the decision to use a *one*-anaphor is made much earlier in the generation process. Section 5 concludes by discussing how this alternate approach might impact both on other aspects of referring expression generation, and on natural language interpretation.

2 Conventional Approaches to Referring Expression Generation

Anaphoric reference to an entity previously mentioned in a discourse can be carried out using any of a number of different strategies: in particular, pronominal anaphora, definite noun phrase anaphora and *one*-anaphora may each be used in appropriate discourse contexts, as demonstrated in examples (1)–(3) respectively.

(1) a. John has a red jumper.
  b. He wears *it* on Sundays.

(2) a. John has a red jumper and a blue cardigan.
  b. He wears *the jumper* on Sundays.

(3) a. John has a red jumper and a blue one.
  b. He wears *the red one* on Sundays.

There is now a well-established body of work in natural language generation that focusses on the problem of generating definite noun phrase anaphora; see Chapter 5 in Reiter and Dale [2000] for a review. Work on the generation of pronominal anaphora is somewhat less developed, with researchers often falling back on some notion of focus as the prime determinant of whether pronominalisation is possible; the major problem here is coming up with an independently motivated notion of what it means to be 'in focus'. The generation of *one*-anaphoric expressions, however, has been virtually ignored, apart from some initial explorations in Davey [1979], Jameson and Wahlster [1982], and Dale [1992, 1995].

A high-level characterisation of the algorithm that underlies much work in referring expression generation is shown in Figure 1. This is deficient in a number of regards: pronouns may be used even if the intended referent is not in focus—see, for example, the centering algorithm of Grosz *et al* [1983]—and a definite noun phrase may be used even if the referent has not been mentioned before, or alternatively its form may be further constrained in some way by the structure of the discourse. However, these complications are not important for our present purposes. The question this paper addresses is as follows: how does the decision to use a *one*-anaphoric expression fit into this kind of algorithm?

3 *One*-Anaphora

3.1 *One*-Anaphora as Syntactic Substitution

The phenomenon of *one*-anaphora is reasonably well discussed in the linguistics literature: in terms of X-bar theory, for example, the pro-form *one* is generally characterised as a substitute for an $\bar{\text{N}}$ constituent (see, for example, Radford [1981:94–95], McCawley [1988:185–186]); and the systemic literature provides some discussion of the nature of *one* as a substitute (see, for example, Halliday and Hasan [1976:89–98]). Although these treatments differ in a number of respects, both characterise effectively the same syntactic constraints on when *one*-anaphora

```
Given an intended referent r:
begin
    if r is in focus then use a pronoun
    elseif r has been mentioned in the discourse already
    then build a definite noun phrase
    else build an initial indefinite reference
end
```

Figure 1: A Skeletal Referring Expression Generation Algorithm

is possible: the *one* form is seen to substitute for a head noun and some number of modifiers of that noun.

For the purposes of natural language generation, we could take this notion of substitution literally: each time we generate a noun phrase structure, we could then compare this against noun phrases in some locally specified discourse context, and then replace any replicated substructure by the form *one*. Assume, for the moment, that a *one*-anaphor always has its antecedent in the previous clause.[1] The generation of *one*-anaphora can then be characterised as follows. Suppose P is a set consisting of the noun phrase structures that appear in the previous clause:

- Given an intended referent r, determine the semantic content needed to identify this referent to the hearer.
- Work out the syntactic structure that realizes this semantic content; call this s.
- Compare s against each $p \in P$, and look for common substructure starting at the head noun and working outwards; replace the largest common substructure found in s by the form *one*.

So, given an antecedent noun phrase as in (4a) and a subsequent noun phrase as in (4b), we can substitute the *one* form to produce (4c), with the *one*-anaphor substituting for the $\bar{\text{N}}$ constituent *mouldy Germanic manuscript.* [2]

(4) a. [a [large [mouldy [Germanic [manuscript $_{\text{N}}]_{\bar{\text{N}}}]_{\bar{\text{N}}}]_{\bar{\text{N}}}]_{\text{NP}}$]
b. [a [small [mouldy [Germanic [manuscript $_{\text{N}}]_{\bar{\text{N}}}]_{\bar{\text{N}}}]_{\bar{\text{N}}}]_{\text{NP}}$]
c. [a [small [one $_{\bar{\text{N}}}]_{\bar{\text{N}}}]_{\text{NP}}$]

There are a number of problems with this approach. First, it sanctions the use of *one*-anaphora where we would want to rule it out on semantic grounds, as in the following constructed example:

(5) a. Do you have any wine bottles?
b. No, but I have a red one.

Second, it rules out *one*-anaphora in cases where the syntactic structures are more distinct, yet we would still want to allow the use of *one*-anaphora, as in the following example:

(6) a. Mary chained her bicycle to a steel fence.
b. Fred chained his to one made of wood.

But quite apart from these concerns (see [Dale 1992:215-230] for a discussion), it also

[1]This is not always true, but the algorithm described here can be trivially extended to deal with other cases.

[2]We will fairly randomly switch between consideration of definite and indefinite *one*-anaphoric forms: for the purposes of the present discussion, any complications introduced by this aspect of discourse status appear to be orthogonal to the issues we are concerned with.

seems a rather wasteful approach. Since the commonality between the antecedent and the anaphor has something to do with shared *semantic* content, why should we go as far as working out the *syntactic* structure required to realise the second NP in order to determine if *one*-anaphora can be used? Syntactic substitution may be an appropriate way to characterise the behaviour of the *one* form when discussing it as a linguistic phenomenon, but that does not mean it should serve as the basis of a generation algorithm.

3.2 *One*-Anaphora as Semantic Substitution

The above objection to the syntactic substitution approach suggests a better solution: look for shared structure at the semantic level. Suppose we have the semantic structure that corresponds to the noun phrase *the red jumper*, and suppose we have gone as far as to generate the semantic content that could be ultimately realised as the noun phrase *the blue jumper*. These semantic structures could be represented as follows:

(7) type(x1, jumper) $\wedge$ colour(x1, red)

(8) type(x2, jumper) $\wedge$ colour(x2, blue)

By identifying what it is that the antecedent NP and the anaphoric NP have in common at the level of semantics, we both avoid unnecessary work in building syntactic structure, and at the same time constrain more correctly the use of *one*-anaphoric forms. This method is elaborated further in [Dale 1992:220–226], and is based on observations made in the work of Webber [1979].

This approach provides us with a way of generating *one*-anaphoric expressions that fits into the general algorithmic structure we sketched in Figure 1; all that is required is that the algorithm maintain a distinction between determining the semantic content of a referring expression and the linguistic realisation of that content, a fairly standard separation useful for other purposes in any case.[3] We then complicate the algorithm to check for the possibility of using a *one*-anaphoric construction once the semantic content has been determined, simply by checking whether there is a replication of semantic content that includes at least the content that would be realised by the head noun. A revised version of the skeletal algorithm is shown in Figure 2.

In suggesting this approach, we have effectively shifted the decision to use *one*-anaphora further back in the generation process, replacing a process of syntactic substitution by one of semantic substitution. In the next section, we argue that we can shift the decision further back still: if we take the stance that *one*-anaphora is typically used to achieve a specific range of discourse functions, that it makes sense to have the discourse planning stage of a generation system impose a requirement that *one*-anaphora should be used when those discourse functions are being realised.

4 Discourse-Driven Generation of *One*-Anaphoric Expressions

4.1 The Functions of *One*-Anaphora in Discourse

Observation suggests that *one*-anaphoric forms are used to achieve particular discourse functions; a common such function, for example, is when a speaker contrasts two entities.[4] It seems reasonable to suppose that, at the discourse planning stage, a generator will already know that it is contrasting two entities; but if the system knows that it is performing a contrast, then at that stage it should already be able to suggest that a *one*-

[3]For example, it allows us to generate *a red jumper* and *a jumper which is red* as variants of the same basic semantic content.

[4]Clearly, an appropriate corpus analysis would determine which particular discourse functions are characteristic of the use of *one*. Just such an analysis is currently underway by Gardiner (forthcoming).

```
Given an intended referent $r$:
begin
    if $r$ is in focus then use a pronoun
    elseif $r$ has been mentioned in the discourse already
    then begin
        build the semantics for a definite noun phrase
        if there is shared structure with a previous noun phrase then elide it
    end
    else begin
        build the semantics for an initial indefinite reference
        if there is shared structure with a previous noun phrase then elide it
    end
end
```

Figure 2: A Revised Skeletal Referring Expression Generation Algorithm

anaphor may be used. In other words: why construct an elaborate mechanism to determine a semantic structure that can be subsequently elided if this means rediscovering something the generator already knew?

The idea that *one*-anaphora is used in the context of particular discourse functions has been noted in the literature before: Dahl [1985] and LuperFoy [1991:114–159] both discuss this aspect of *one*-anaphora at some length. LuperFoy's observations are closest to those that lie behind the view taken here. She suggests that uses of *one*-anaphoric forms correspond to three particular discourse functions: to contrast two sets of individuals, to denote a representative sample of a set introduced by the antecedent, and to refer to a new specimen of a type that is salient in the discourse; examples of each of these categories are provided in (9)–(11) respectively:

(9) a. John has a magenta Capri.
 b. Robert has *a reef-green one.*

(10) a. John has several cars.
 b. *The smallest one* is a Capri.

(11) a. John has several old cars.
 b. Mary wants to buy him *a new one.*

In the terms of Rhetorical Structure Theory [Mann and Thompson 1987], the discourse function in (9) is one of CONTRAST, and those in (10) and (11) are instances of the ELABORATION relation.

4.2 How We Might Integrate *One*-Anaphora in Text Planning

We are concerned in the first instance with the monologic case, where both the sentence containing the *one*-anaphora and the sentence containing its antecedent are spoken by the same conversational participant; as will become clear, a quite separate explanation is likley to be required for dialogic uses of *one*.

In the sample discourses just presented, it seems plausible to suggest that the two sentences are 'spoken as pairs'. In (9), the speaker utters the two sentences precisely in order to draw a contrast; in (10) and (11), the second sentence is only a coherent contribution to a discourse given the background provided by the first sentence.

In a natural language generation system which performs text planning, we take the view that the contrast or elaboration that is being performed is the most important issue; the particular linguistic expressions constructed are subsidiary to these aims. Viewed in this way, it makes sense for the text planner to PRESELECT some of the linguistic features of the utterances to be produced when the discourse relation has been decided upon.[5] Clearly there are other forms of contrast than those realised by means of *one*-anaphors, and a fully-fleshed out model of how this preselection mechanism might work will require finding an appropriate level of abstraction for expressing 'linguistically-realised contrast'; however, for present purposes we can focus on instances of contrast where it is similar entities that are being contrasted, and assume for simplicity that *one*-anaphora is the only appropriate contrastive device available.

Any text planning component has to decide when it wants to use rhetorical devices such as contrast. The proposal here is that, when such a goal has been selected for whatever reason, then, provided some additional constraints are met, the text planner can already at that point determine that specific linguistic forms should be used. In effect, the choice of a specific discourse relation brings with it linguistic consequences. This is entirely plausible where discourse connectives are concerned: a decision to use, for example, a relationship of CAUSE might lead automatically to the decision to use the discourse connective *because*. Here, we are extending this idea to cover also elements within the expression of the propositions to be conveyed. The present case under discussion is shown schematically in Figure 3, where a desire to use a CONTRAST relation, combined with a particular configuration of knowledge in the knowledge base, results in the use of a specific rhetorical structure with some prespecified lexical content. In this case, the constraints on the configuration of knowledge are that the two entities x_1 and x_2 share the same

[5]The use of the term 'preselect', a term from work in systemic approaches to generation, is deliberate. What we are arguing for here amounts to an interstratal preselection of lexicogrammatical features in the sense of Matthiessen and Bateman [1991:62–65].

$$\begin{bmatrix} \text{relation:} & \text{contrast} & \\ \text{nucleus:} & \begin{bmatrix} \text{s:} & \begin{bmatrix} \text{np:} & \ldots \\ \text{vp:} & \begin{bmatrix} \text{v:} & \ldots \\ \text{np:} & \begin{bmatrix} \text{index:} & x_1 \\ \text{syn:} & \begin{bmatrix} \text{det:} & \ldots \\ \text{adj:} & \mathsf{A}_1 \\ \text{n:} & \mathsf{T} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{satellite:} & \begin{bmatrix} \text{s:} & \begin{bmatrix} \text{np:} & \ldots \\ \text{vp:} & \begin{bmatrix} \text{v:} & \ldots \\ \text{np:} & \begin{bmatrix} \text{index:} & x_2 \\ \text{syn:} & \begin{bmatrix} \text{det:} & \ldots \\ \text{adj:} & \mathsf{A}_2 \\ \text{n:} & \begin{bmatrix} \text{lex:} & \text{one} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3: A schematic discourse structure that preselects lexical material

semantic type but have differing values for other attributes, here expressed by the adjectives A_1 and A_2; precisely the circumstances under which *one*-anaphora is possible.

Clearly the picture is considerably more complicated than this simple sketch implies, but the general idea should be clear. As suggested above, what this view does is to push the decision to use a *one*-anaphoric expression further back still in the generation process. This sites the decision in a far more appropriate place: deciding when a contrast should be made is a much larger question that must be faced by any text planning system. Ultimately, the view taken here is that contrast is just one device that we use to produce coherent discourse: one way of characterising the general problem for a text planner is as the decision of what to say next, and here notions like topic maintenance and topic chaining are crucially important. Contrasting two clusters of information stored in a knowledge base is just another of these associative devices that can be used to build a coherent text on the basis of relations that reside in the underlying knowledge base.

5 Conclusions and Future Work

We have argued that *one*-anaphora is best viewed as a linguistic phenomenon that is a natural consequence of the speaker's choice to use specific subject-matter discourse relations, and that a consequence of this is that the decision to use *one*-anaphora should, at least in part, be determined at the level of discourse planning. We have sketched how this might work in the case of contrastive uses of *one*, but a similar story can be told for the set-elaborative function.

At the outset, we asked how the generation of *one*-anaphora could be integrated into existing referring expression generation algorithms. These algorithms assume that they are given some symbol that corresponds to the intended referent, and then attempt to determine what content should be used to identify this intended referent. This model is incompatible with the approach proposed here, since the approach we have argued for lacks a distinct stage in the processing where the intended referent is only indicated by some internal symbol. In order to integrate the generation of *one*-anaphora into conventional generation algorithms, the assumption that the referring expression generator is given nothing more to work with than the symbol that corresponds to the intended referent has to be abandoned, and the bandwidth of communication between the discourse planner and the referring expression generator increased: ideally, the referring expression generator is told not only what the intended referent is, but also what its function in the discourse is.

There is some precedent for this idea. McDonald's [1980] work on referring expression generation within MUMBLE includes a facility whereby the expert system driving the generator can specify that a message element (i.e., an internal symbol corresponding to the intended referent) is 'ontologically of a sort that cannot be pronominalized' [1980:217]: this allows the expert system to specify that some information has to be expressed for descriptive, rather than purely referential, purposes. A similar broadening of the bandwidth is visible in McKeown's TEXT [McKeown 1985], where the text planning component can indicate to the linguistic realisation component that a particular entity is the focus of the utterance, resulting in pronominalisation; and the same idea finds expression in the use of the CENTRE attribute in Dale's EPICURE [1992:170–171]. The present work suggests that these devices can be seen as instances of a more general mechanism where the discourse purpose of a referring expression plays a role in how that referring expression is best realised. Above, we have discussed one specific discourse function, which we might characterise more precisely as CONTRAST-TWO-SIMILAR-ENTITIES; other instances of the use of *one* would be characterised by the discourse function SELECT-ELEMENT-FROM-MENTIONED-SET. The same idea, however, can be used to provide a new way of thinking about existing well-explored reference tasks: so, for example, in appropriate discourse contexts, pronominalisation may be an automatic consequence of the discourse functions MAINTAIN-AS-FOCUS and SHIFT-INTO-FOCUS; initial reference might be best thought of as a consequence of the discourse function INTRODUCE-ENTITY; and different instances of subsequent reference might be cases of either DISTINGUISH-ENTITY or ATTRIBUTE-ADDITIONAL-INFORMATION, or even combinations of both.

Further work is required in order to determine how best to rearrange generation architectures to integrate these observations. By abandoning traditional architectural divisions into pipelined components, systems based on systemic functional grammar (see, for example, [Matthiessen and Bateman 1991]) already allow sufficient flexibility to incorporate the mechanisms discussed here. However, the absence of distinct processing modules with well-defined interfaces between them is generally considered to make it more difficult to build practical systems which can be easily re-used and maintained. A question for further research is whether, taking the observations of this paper into account, we can characterise the required interactions between referring expression generation and other aspects of the generation task in such a way that modular systems can be built.

An additional interesting direction that is opened up by this view is that of how we might revise our models of natural language analysis to take account of the interstratal relationships between discourse planning elements and surface forms. If, for example, we can characterise the generation of a pronominal form as discourse-planning construct that has as its base a discourse function of MAINTAIN-AS-FOCUS, then we may also be able to use such a multi-level construct when interpreting a pronoun: the idea here would be that, instead of using a more traditional level-by-level analysis (syntactic analyis, then semantic analysis, followed by interpretation in context), we can hypothesis information at all of these higher levels simultaneously on the basis of the presence of the surface form. Of course, this is only a sketch, and there is significant work to be done in fleshing this out; however, the basic idea offers a novel way of thinking about both language analysis and language generation.

References

D Dahl [1985] *The Structure and Function of One-Anaphora in English.* Indiana University Linguistics Club.

R Dale [1992] *Generating Referring Expressions.* MIT Press.

A Davey [1978] *Discourse Production.* Edinburgh University Press.

M Gardiner [forthcoming] *Identifying and resolving one-anaphora.* Honours Thesis, Centre for Language Technology, Macquarie University.

B Grosz, A Joshi and S Weinstein [1983] *Providing a Unified Account of Definite Noun Phrases in Discourse.* In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, Massachusetts, 15th–17th June 1983.

M Halliday and R Hasan [1976] *Cohesion in English.* Longman.

E Hovy [1991] *Approaches to the Planning of Coherent Text.* Pages 83–102 in C Paris, W Swartout and W Mann (eds), *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers.

S LuperFoy [1991] Discourse Pegs: A Computational Analysis of Context-Dependent Referring Expressions. PhD Thesis, University of Texas at Austin.

J McCawley [1988] *The Syntactic Phenomena of English*, Volume 1. The University of Chicago Press.

D McDonald [1980] Natural Language Generation as a Process of Decision-Making Under Constraints. PhD Thesis, MIT.

K McKeown [1985] *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press.

W Mann and S Thompson [1987] Rhetorical Structure Theory: A Theory of Text Organisation. USC/
Information Sciences Institute Technical Report RS–87–190.

C Matthiessen and J Bateman [1991] *Text-Generation and Systemic Functional Linguistics.* Pinter.

A Radford [1981] *Transformational Syntax.* Cambridge University Press.

E Reiter [1990] Generating Appropriate Natural Language Object Descriptions. PhD Thesis, Harvard University.

E Reiter and R Dale [1992] A Fast Algorithm for the Generation of Referring Expressions. In *Proceedings of Coling–92*, Nantes, France.

E Reiter and R Dale [2000] *Building Natural Language Generation Systems.* Cambridge University Press.

P Sibun [1991] Locally Organized Text Generation. COINS Technical Report 91–73, Department of Computer and Information Science, University of Massachusetts.

H Thompson [1977] Strategy and Tactics in Language Production. In *Papers from the Thirteenth Regional Meeting of the Chicago Linguistics Society*, Chicago.

B Webber [1979] *A Formal Approach to Discourse Anaphora.* Garland Press.

Application of Search Algorithms to Natural Language Processing

Takeshi Matsumoto
School of Informatics and Engineering
Flinders University of South Australia
takeshi.matsumoto@flinders.edu.au

David M. W. Powers
School of Informatics and Engineering
Flinders University of South Australia
powers@infoeng.flinders.edu.au

Geoff Jarrad
Business Intelligence Group
CSIRO ICT centre
geoff.jarrad@csiro.au

Abstract

Currently, the most common technique for Natural Language parsing is done by using pattern matching through references to a database with the aid of grammatical structures models. But the huge variety of linguistical syntax and semantics means that accurate real time analysis is very difficult. We investigate several optimisation approaches to reduce the search space for finding an accurate parse of a sentence where individual words can have multiple possible syntactic categories, and categories and phrases can combine together in different ways. The algorithms we consider include mechanisms for ordering that reduce the search cost without loss of completeness or accuracy as well as mechanisms that prune the space and may result in eliminating valid parses or returning a suboptimal as the best parse. We discuss the development and benchmarking of the existing and proposed algorithms in terms of accuracy, search space and parse time. Speed up of an order of magnitude was achieved without loss of completeness, whilst decrease of over two orders of magnitude was achieved in the search space. A further order of magnitude reduction of both time and search space was achieved at the expense of some loss of accuracy in finding the most probable parse.

1 Introduction

The complexity and the sizes of the lexical databases and grammatical rules contribute to most of the behaviour of Natural Language parsers. By increasing the sizes of the database or including a more complex set of grammatical rules, the parser is able to handle the parsing of more complex sentences or is able to include more accurate information to the parsed sentences, but the introduction of these results in a more complex parsing procedure and the capability to compute for more cases is necessary for the parser. Even without the extended database or rules, parsing of long sentences is often avoided due to the extremely large amount of different possibilities in parsing the sentence.

To counteract the increase in the parse time form the application of complex grammatical rules, we explore the effects of applying search algorithms to a parser to reduce the search space and hence enhance the parsing speed. To measure the accuracy of the parse, we use a simple scoring system derived from the probability that a particular structure would exist. This scoring system does not always parse the sentence correctly, but it provides a good indication of the likeliness of the structure from a statistical point of view.

The purpose of the project is to provide a faster way of parsing sentences without losing the effect of grammatical structures, or the semantic and syntactic information that have been applied to or extracted from the parser. These areas being the key focus of most research done in NLP and will continue to increase in complexity in the future.

2 Parsing

The parser we are using was the probabilistic, lexicalised combinatory, categorical grammar (PLCCG) parser implemented by the CSIRO[1] (Jarrad et al., 2003) that incorporates a bottom-up search strategy. In the training stage, the parser builds up a statistical model of the grammatical structure by learning from a manually parsed corpus, which is used to assign the possible categories and the probabilities of the particular category for a word, and also the probabilities associated with the actual combination of two structures. The CCG[2] (Steedman, 1996) incorporated in the parser defines the rules and methods used in the combination stage of the parser, and implements an extended set of the standard CCG combinators (Jarrad et al., 2003) that makes the grammar more flexible. The nature in which a combination occurs is very much like using the link grammar rules to combine between the different states.

Initially, the individual words are given a set of potential categories that it has seen for the particular word in the training corpus. Due to the varieties in the training data and the increase in freedom gained from the extended grammar, some words are given a huge set of potential categories. This creates a more robust grammar, which can handle the parsing of complex sentences, but also contributes to an explosion in the search space. If the particular word was not seen in the training corpus, a lexical database called WordNet (Miller et al., 1993) is used to assign the possible categories for the word. This is done by extracting the part of speech (POS) tags for the unknown word and assigning all the possible categories for that POS to the word. Finally, if the word was not found in WordNet, the set of all possible categories are assigned to the unfamiliar word with the probability of the category being the frequency of the category form the entire training corpus. The difference in the number of initial categories for a word can be enormous, ranging from one to over one thousand.

The probability scores for the states are derived from the combination of 3 transition probabilities, the word category transition, categorial transition, and the lexical transition. The parser uses these probabilities to derive the scores of a parse to find the most probable parse, which is derived by exhaustively combining all possible states for the parsing sentence. The approach in which this is done is very similar to the chart parser (Charniak, 1993). This eventually results in the formation of a state combing all words in the sentence, which we call the terminal state. The scores of all the terminal states are compared and the parser returns the parse tree structure for the most probable state. If there are multiple states that are equivalent in how it was structured, the parser keeps the state with the higher probability. For duplicate scores, the first one it encounters is kept.

The task of finding all of the possible combinations is almost as difficult as the travelling salesman problem. The search space expands as the third power of the words, but due to the fact that states can only combine with adjacent states, some reduction in the search space occurs automatically. But on top of the possible combinations of the sequences of words, there is a squared factor for the number of possible categories each combination would need to consider, which results in the model:

$$\sum_{i=1}^{l-1} \sum_{j=i}^{l-1} \sum_{k=j+1}^{l} N_{i,j} \cdot N_{j+1,k}$$

Where i, j, and k represents the starting position of the left sequence, ending position of the left sequence, and the ending position of the right sequence, respectively. l represents the number of words in the sentence and $N_{i,j}$ represents the number of states for the sequence between i and j.

The above model clearly indicates that the task of parsing, especially when the number of categories for a word can be of the order of several hundreds, is a lengthy task. This value can reach up to several millions even on sentences with less than 10 words. Hence the need for a search algorithm that would prune the search space without any loss of accuracy.

3 Optimal-Search Algorithm

The major goal of this project was to explore alternative standard and novel algorithms that were appropriate to the task and could relatively easily be slotted into the existing parser framework. The

[1] Commonwealth Scientific and Industrial Research Organisation

[2] Combinatory Categorial Grammar

kind of algorithms and optimisations that are reasonable is tightly constrained by the nature of the CCG model and the PLCCG implementation. Another major constraint of the algorithm is one that is often ignored, which is the overhead in the execution of the algorithms. This factor plays an equally important role in the search problem, but has often been ignored due to the increase in the hardware performance rate. The algorithmic design was modularised, so that an easy switching of the algorithm could be done with a uniform interface to the rest of the original parser. This meant that the algorithm relied on some of the existing structure of the parser, which was the cause of some limitations in the algorithms and is an area that could be modified in the future to further increase the efficiency of the parser.

The first algorithm that was considered was Adaptive probing (Wheeler, 2001) and this was tested on a subset of the problem by simulation using a toy language (Kilby, 2002). This algorithm was considered due to the gain in search speed seen in the simplified search problem, but was rejected due mainly to the random nature of the search, which means that an exhaustive search was necessary to provide the most probable parse.

The first enhancement was to apply a different ordering of the combinations to allow the fast build up of the relevant sections of the parse tree. By ranking the states in order of their probabilities, the parse tree was built up in such a way that the most probable state in the tree was considered first. Due to the randomised build up of the parse tree in terms of extension of the branches in the search tree, the algorithm had to include an indicator to allow the extension of branches from nodes, even after it had been used to construct its children states already. This backtracking mechanism was implemented using a list containing all of the states, which was divided into two sections. A pointer into the list indicated the division point between the two sections, one of which contained all of the states that had been used to combine with other states, and the other section contained all of the states that had not been used to combine with other states. Whenever a state was used to combine with other states, it was placed in the used section and the states that resulted from the combination were placed in the non-explored section. This divided list ensured that no two same combinations would ever occur more than once.

To apply the ranked ordering, the list was maintained in a sorted manner by their probability scores and the pointer simply moved along the list, as more states were used to combine with other states. The state being pointed to by the pointer, which was the state being used to combine with other states of higher scores, was called the pivot state. By combining the pivot state with states of higher scores, the algorithm guaranteed that resulting state of the combination would be equal or lower scored than the pivot state. This allowed for a simple algorithm for maintaining the ordered list. The ranking algorithm is essentially embodied by the following pseudo-code:

```
1. Populate the list with every state for every
     word.
2. Sort the list by their probability scores.
3. Set pointer at the first state in the list.
4. While the list contains un-combined states:
5.   Set pivot as the next most probable state.
6.   Return if pivot state is a terminal state.
7.   Combine pivot with all adjacent states
       with higher probability.
8.   Insertion sort all newly created states in
       to the list.
9. Return failure
```

With the application of this ordering, the algorithm allowed for early termination of the search, since the newly created states (being of equal or lesser probability) must be inserted below the pivot state due to the cascading effect of the product of the probability. Any terminal state found later would have a lower probability than the first one that was found, so the algorithm guarantees the retrieval of the most probable state without having to exhaustively search all possible combinations.

By only using a single list to maintain all possible derivation of the states, traversals and maintenance of the ordering of the list used up a lot of valuable time. To counteract this, we re-introduce a charting behaviour as the second improvement to the algorithm. We implemented a table, called the indexed table, in which all the states that were in the used section were placed, rather than keeping them in the same list. The table also grouped together the states that occupied the same starting and ending positions, to simplify the decision process in determining which states were adjacent to the pivot state. The ranked list was replaced by a table, which we called the sorted table that handled the push and pop manipulations to simplify and to modularise the algorithm for future use.

The third major step involved the use of a critical score, which is the score of the currently most probable terminal state in the sorted table. By not operating on states that are going to produce a lower probability than the critical score, it allowed for a large pruning of the search tree, weeding out states with very low probability that would not contribute to the most probable terminal state. The algorithm also provides a pre-processing stage before a combination between states took place, which contributed to a little overhead, but managed to cut down the amount of unnecessary combinations and avoided the lengthy combination stage of two states.

The scoring system, as it stood, meant that combined states of large length would have a very low score, even if they consisted of very probable sub-structures. There was a necessity to allow larger sized states a better score to indicate their higher desirability. The next major step in the evolution of the algorithm was to alter the scoring system to allow larger sized states higher ranking than by the use of the raw probability scores. This was achieved by normalising the scores to the most probable scores of the corresponding positions of the states and hence altered the ranking system so that states that occupied different sections in the sentence were compared relative to other states that occupied the same sequence of words. The scores of a potential perfect combination of the most probable states for each word were used to derive the normalisation scores for the particular sequence. This is not the most accurate way of determining the normalisation scores, but it provided an efficient way to change to ordering while not causing too much overheads in the pre-processing stage. The normalisation scores and the scores used for ranking are derived by:

$$S_{i,j}^{normal} = \prod_{k=i}^{j} S_{k,k}^{\max}$$

$$\begin{aligned} S_{i,j}^{rank} &= S_{0,i-1}^{normal} \cdot S_{i,j} \cdot S_{j+1,l}^{normal} \\ &= S_{0,l}^{normal} \cdot S_{i,j} \Big/ S_{i,j}^{normal} \end{aligned}$$

In the above model, *i* and *j* represents the starting and ending indexes of the state and *l* represents the length of the sentence. $S_{i,j}^{normal}$ represents the normalization score for the sequence in the range between *i* and *j*, $S_{k,k}^{max}$ represents the score of the most probable states for word at *k*. $S_{i,j}^{rank}$ represents the score used for ranking, but it also represents the heuristical score of the state. $S_{i,j}$ represents the raw probability score of the particular sequence which starts and ends at positions *i* and *j*. Note that the $S_{0,l}^{normal}$ is a constant for the same sentence and can be factored out for the purpose of ranking, which gives:

$$S_{i,j}^{rank} = S_{i,j} \Big/ S_{i,j}^{normal}$$

The combined algorithm still maintains the retrieval of the parse tree with the same probabilistic score as the exhaustive algorithm, but has managed to prune a very large section of the search tree without creating too much overhead in the execution of the algorithm. The pseudo–code for this new algorithm is:

```
1.  Construct the normalisation mapping.
2.  Initialise the critical score to zero.
3.  Populate and sort the sorted table with all
      states for all words using the normalised
      scores.
4.  Remove the most probable state and insert
      into the indexed table.
5.  While the sorted table contains un-
      combined states:
6.      Remove the most probable from the
          sorted table as the pivot.
7.      Return if the pivot is a terminal state.
8.      Combine pivot with all adjacent states in
          the indexed table that don't fall below
          the critical score.
9.      For every state that has been created:
10.         Adjust the critical score if the produced
              state is a terminal state and the score
              is better.
11.         Insert the created states into the sorted
              table with the normalised score.
12.     Insert the pivot into the indexed table.
13. Return failure.
```

We also investigated alternative algorithms that included more pruning in the search tree, and also the effects of prematurely ending the search when an approximate result was found. We experimented with ideas like pruning lower scored states at the start of the algorithm (beam search), approximating the correct parse to be the first terminal state it found, and applying a different priority system that encouraged the build up of larger sized states without first building up the sub-structures. The beam search had the same effects to the parser as a reduced set of categories and combinators, in that, some valid sentences could not be parsed because of the reduced amount of ways in forming the valid parse. This is a very common approach taken to optimise a searching task (Goodman, 1997), but was not the desired approach for this project since the task of the algorithm was to find the most probable parse for the sentence.

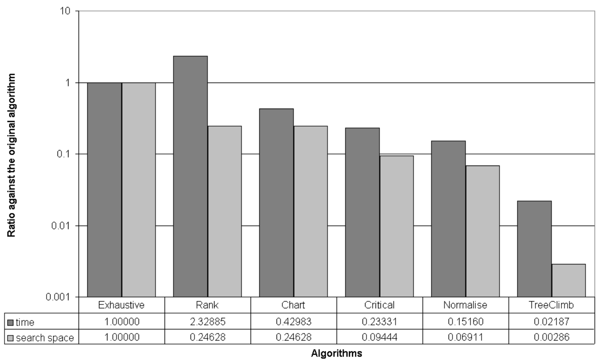
By terminating the algorithm prematurely, the parser sometimes retrieved the non-optimal result and also did not contribute much to the reduction of the parsing time, due to the improved ordering of the search algorithms.

By re-ordering the search, so that the build up of larger sized states were prioritised, the effects of the ordering by their scores were lost and hence the algorithm had to either exhaustively search all combinations to determine the most probable parse, or it had to end the algorithm after the production of some terminal state was made. This did not guarantee the retrieval of the most probable parse and it also meant that some unlikely combinations that could have been avoided by the ranking had to be done.

When implementing most of the experimental algorithms, some of the core structure to the algorithm had to be modified, but an interesting algorithm was discovered in the process. This was the product of the beam search and the prioritising of larger states, which we called the tree-climbing algorithm. The beam search stage, which we called the seeding stage, involved building of the parse with only the most probable state for each of the word, and the tree-climb approach, which was applied in the subsequent stage, resulted in an algorithm that was faster than the combined optimal algorithm, but was not as accurate when it came to the retrieval of the most probable parse. However, the proportionality of the incorrect parse was significantly lower than the application of just the beam search or the tree-climb algorithm. The tree-climb approach was not attractive in terms of both parsing accuracy and time in the cases where the sub-structures had to be built up first when it was applied by itself, but the seeding stage constructed the majority of the necessary sub-structures in the search tree, and hence allowed the tree climb algorithm to connect up the un-combined sections and quickly form a parse for a sentence. Although this algorithm did not guarantee the retrieval of the most probable parse, it provided alternative points of view in the relevance of the scoring system which was used to determine the 'correct parse' of the sentence; some parses which were retrieved were structured more similarly to the humanly evaluated parse than the most probable parse, even though they were assigned lower probabilities.

4 Results

The algorithms were trained and tested on both the Susanne corpus and the Penn Treebank corpus (Mitchell et al., 1992), approximately 95% of each was included in the training sets (sections 02 to 21 for the Penn Treebank) and a randomly chosen subset from the rest of the corpus was used for the testing sets (section 23 for the Penn Treebank corpus). This corresponded to 50 sentences in the Susanne corpus and just over 580 sentences in the Penn Treebank corpus. The major difference between the two corpora is the number of possible categories it contains. Where the Susanne corpus contains just over 500 categories, the Penn Treebank corpus contains over 1200 categories it can assign to each of the words. The two corpora were used to test the performance of the developed algorithms; hence the parsing accuracy is not the intended matter being evaluated here.



	Exhaustive	Rank	Chart	Critical	Normalise	TreeClimb
▩ time	1.00000	2.32885	0.42983	0.23331	0.15160	0.02187
▢ search space	1.00000	0.24628	0.24628	0.09444	0.06911	0.00286

Figure 1: Benchmarked logarithmic plot comparing the exhaustive algorithms to the developed algorithms on the Susanne corpus.

The PLCCG parser and the developed algorithms were implemented on Python, the reason being that the parser is still under development in the areas of syntactic and semantic accuracy.

The progressive increments to the proposed algorithm all contributed to large areas of the search space being pruned, but due to the overheads in the execution of the algorithm, some of the benefits were not as much as first expected.

Figure 1 indicates the ratio differences between the original exhaustive algorithm and the proposed algorithms on the Susanne corpus. The results from the Penn Treebank corpus are not shown, since we were unable to obtain the results for some algorithms due to memory problems. The left column indicates the parsing time and the right column indicates the amount of search space it explored, or the number of combinations made during the parse. The difference between the two indicates a rough estimate of the overhead in the execution of the algorithm compared to the original algorithm. As the plot indicates, the parse time of the ranked algorithm was excessive. The overhead in determining the adjacent states contributed to most of the parse time and resulted in a worse parse time, even though it was only exploring a quarter of the search space.

After the charting was implemented, the benefits of the algorithms became more apparent, even through it was still exploring the same search space. The reduction of the search space by 75% from the use of the ranked ordering indicates that most of the categories assigned to the initial words did not make much contribution, due to the rareness of its own category and the corresponding derived states.

The inclusion of the critical score made another dramatic reduction in the search space, indicating that a lot of unnecessary searching was occurring after the terminal state was produced, which could be carefully pruned out without affecting the final result. The proportionality between the search time and the search space increased with the inclusion of the critical value, which was contributed by the overheads in the pre-processing stage before the combination between the states occurred.

The use of the normalised scores contributed to yet another reduction in the parse time and search space. This algorithm did not make as much use of the critical value compared to the raw critical algorithm due to better ordering of the states, but since the normalisation scores are not the perfect representation of the relative scores to each position, which is impossible to predict, the critical value still plays an important role in the algorithm. This algorithm introduces extra processing to calculate the normalised scores and to re-order the states with the same scores, but the overheads is still a lot less than the raw critical scored algorithm, due to the repeated pre-processing overheads.

The experimental tree-climb algorithm result seen on the far right shows an impressive parse time and huge reduction in the search space, but has slight inaccuracies parse compared to the other algorithms, which can be seen in Table 1.

	Exhaustive	Optimal	Suboptimal
Susanne			(%)
Parse time	100.0	15.2	1.7
Search space	100.0	6.9	0.3
Most probable	100.0	100.0	84.0
Penn Treebank			(%)
Parse time	100.0	10.4	0.7
Search space	100.0	4.7	0.1
Most probable	100.0	100.0	66.7

Table 1: Statistics of parsing of the optimal and suboptimal algorithms for both the Susanne and Penn Treebank corpora.

The parse time and the search space are represented as the proportionality compared to the exhaustive algorithm and the percentage that the algorithm retrieved the most probable parse is indicated in the last row. The optimal algorithm is the combined algorithm of all the algorithms that provided benefits to the parsing speed without the loss of accuracy and the suboptimal algorithm is the tree-climb algorithm, which provided a fastest and also a reasonably accurate result from all tested suboptimal algorithms.

The results from the 2 corpora indicate similar trends in the characteristics of the algorithms, which indicate a consistent improvement from the application of the algorithms. The parse time and the search space showed a bigger improvement from the larger Penn Treebank corpus, even though there is over twice the number of categories to choose from. This is suspected to be the fact that more trivial sentences exists within the Penn Treebank testing set. Another contributing factor to this is the fact that the training set is a lot larger in the

Penn Treebank test. This means that the algorithm does not need to look up unknown words from WordNet or spend time assigning all possible categories for the word.

The optimal search algorithm returns the most probable parse tree, but sometimes varied in the tagging and bracketing of the parse due to the cases when multiple parses have the same probability. The tree-climb algorithm's performance in the accuracy domain is relatively poor, but some of the loss in the accuracy can be recovered by altering the amount of states used in the seeding stage. However, because the algorithm loses track of the ranking of the states, the algorithm must exhaustively combine all states to determine the most probable parse.

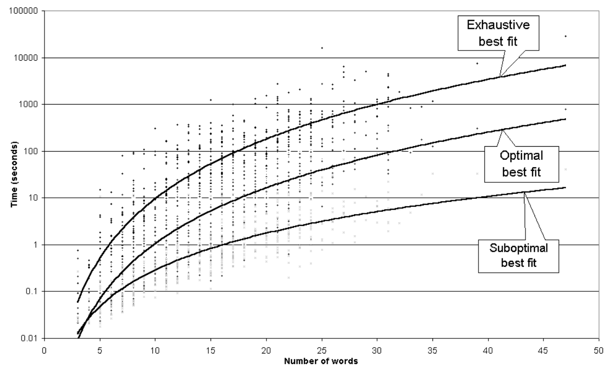


Figure 2: Number of words in the sentence versus parsing time on the Penn Treebank corpus for the exhaustive, optimal and the suboptimal algorithm.

On a corpus based comparison, it is fairly easy to see the improvements of the developed algorithms, but for the task of NLP, it is probably more important to look at a per sentence comparison, especially if it is in an environment where human interaction is required. Figure 2 indicates the relationship between the parsing time and the number of words in the sentence for the exhaustive, optimal and the suboptimal search algorithms. There is a huge reduction in the parse time from the algorithm with the optimal algorithm, and an even greater reduction from the suboptimal algorithm. Both these algorithms possesses another great feature in that the exponential coefficient factor for the parse time is a lot less than the exhaustive algorithm. This means that the algorithm works more efficiently with longer sentences, but the plot still indicates that the new algorithms are still better than the exhaustive algorithm for short sentences, even with the extra overheads from various forms of initialisation.

The long parsing times are the consequences of using a scripting language for the development and testing of the parser. The results should reduce by a factor of several tens or even hundreds if the parser was implemented on a natively compilable language.

Figure 3 describes the overall efficiency of the algorithms, which displays 4 different dimensions about the algorithms. The first is the linear slope seen in all the algorithms. This indicates that the non-searching processes in the individual algorithms (overheads), like the initialisation stage do not contribute greatly to the parse time, with the exception of the set of plots around the 10 second range, which has deviated from the other results. This is due to the extra time taken for the algorithms to fetch the relevant categories form WordNet and also the assignment of all possible categories. This is not apparent in the exhaustive algorithm, because they perform a lot more combinations if the number of categories assigned to each of the words is large, where as the optimal and the suboptimal algorithms does not take most of them into account. The actual slope of the plots indicates the sizes of the coefficient of the relationship, shown more clearly in Figure 2.

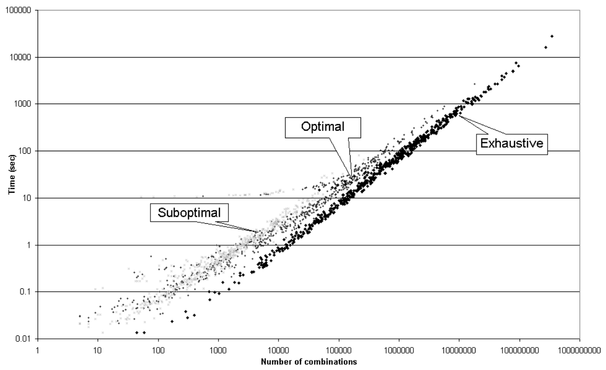


Figure 3: Efficiency of the algorithms measured on the Penn Treebank corpus.

The second dimension is the y-intercepts of the plots, which describes the efficiency of the execution of the algorithm. The smaller the y-intercept, the more efficient it is in executing each combination. The exhaustive algorithm clearly outperforming the others, due to the simple way it needs to be implemented. Due to the large overhead in applying the suboptimal algorithm, there is a large overhead in the algorithm, meaning that if the states of a very low probability had to be used to produce a terminal state, this algorithm would run the slowest.

The third dimension is the spanning distance between the plots, which indicates the size of the exponential coefficient. The longer the distance between the quicker parses and the longer parses, the larger the exponential coefficient it has. The effect of which can be seen more clearly on Figure 2. The fourth and the final dimension is the average speed of the parse time, which is indicated by the average height of the points.

5 Conclusion and Future Work

Unlike most modern search algorithms that take advantage of the continuously increasing processing power of the modern day computers and hence lose elegance in the search technique, the developed search algorithm allows for the retrieval of the best possible solution in a very efficient manner while also taking into account of the overheads involved in execution of the algorithm. The implementation of the algorithm as the searching mechanism to find the most probable parse for the target parser has dramatically reduced the parsing time required to retrieve the same result as an exhaustive search mechanism.

The characteristics of the algorithm has the potential to be converted into a simple chunk parser, which is sometimes enough to extract the relevant information from the sentences. The proposed algorithm encourages the quick build up of subparses, rather than the linear build-up algorithm of the exhaustive algorithms, hence the order in which the combination occurs allows for the splitting of the sentence into sections or chunks by early termination of the algorithm.

The tree-climb algorithm needs further investigation, as the algorithm may possesses characteristics which may end up being more beneficial to the accuracy of the parser. This is due to the fact that the most probable parse is not always the correct parse. Further investigation techniques might include getting the algorithm to find multiple solutions before it is returned, or to measure the accuracy after altering the amount of states used in the seeding stage. Primitive experiments done on adjusting the seeding amount has decreased the error rate, but further tests are required to understand the effects on this.

By modifying the probability scores to include information on things like the syntactic and semantic context to provide a better indication of the grammar which will provide a better scoring system, the parser should be able to provide better results and still rely on the developed optimal algorithm to retrieve the most probable parse. This also means that the algorithm is generic enough to be applied to other kinds of search problems. The 4 main implemented techniques; ranking the nodes in the search tree to allow for early search termination, using charting to avoid processing of unwanted search space, applying the critical point scores and a quick pre-processing stage to avoid lengthy computation, and the use normalised scores to provide a better heuristical indication of the node being searched all provide vital ways to reduce the search space of the problem.

Acknowledgement

This project was supported in part by a scholarship from the CSIRO.

References

Eugene Charniak. 1993. *Statistical Language Learning*. MIT Press, Cambridge, England.

Jushua Goodman. 1997. *Global Thresholding and Multiple-Pass Parsing*. Proc. EMNLP-2.

Geoff Jarrad, Simon Williams, and Daniel McMichael. 2003. *A Framework for Total Parsing*. CMIS technical report 03/10.

Phil Kilby. 2002. *Applying Adaptive Probing to Achieve Fast Parsing*. Project proposal paper.

Mitchell P. Marcus, Beatrice Santorini, Mary A. Marcinkiewicz. 1992. *Building a large annotated corpus of English: the Penn Treebank*. http://www.cis.upenn.edu/~treebank/home.html.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1993. *Introduction to WordNet: An On-line Lexical Database*. http://www.cogsci.princeton.edu/~wn/.

Mark Steedman. 1996. *A Very Short Introduction to CCG*. ftp://ftp.cis.upenn.edu/pub/steedman/ccg/ccgintro.ps.gz

Ruml Wheeler. 2001. *Incomplete Tree Search using Adaptive Probing*. Proc. IJCAI-01.

An Empirical Study for Generating Zero Pronoun in Korean based on Cost-based Centering Model

Ji-Eun Roh
Div. of Electrical
and Computer Engineering
Pohang University of Science and Technology and Advanced Information Technology Research Center (AITrc)
San 31, Hyoja-dong, Nam-gu, Pohang, 790-784, R. of KOREA
jeroh@postech.ac.kr
fax: +82-54-279-5699

Jong-Hyeok Lee
Div. of Electrical
and Computer Engineering
Pohang University of Science and Technology and Advanced Information Technology Research Center (AITrc)
San 31, Hyoja-dong, Nam-gu, Pohang, 790-784, R. of KOREA
jhlee@postech.ac.kr
fax: +82-54-279-5699

Abstract

In Korean, in order to generate a coherent text, a redundantly prominent noun should be replaced by a non-zero pronoun or zero pronoun. Otherwise, the text becomes unnatural. Specifically, a redundant noun in Korean is frequently omitted while a redundant noun in English is replaced by a pronoun. This paper proposes a generation algorithm of the zero pronoun, using a *Cost-based Centering Model* which considers the inference cost. For an objective evaluation of our algorithm, we collected 87 texts from three genres, and manually recovered the omitted elements. Using the collected texts, we verify that our algorithm is well defined to explain the phenomenon of the zero pronoun in Korean. We also show that the proposed approach resolves both the over-generation of the zero pronoun in *Continue* and its under-generation in other transitions in terms of Centering.

1 Introduction

Text generation is the process of producing comprehensible texts in natural languages from non-linguistic representation of information. To generate a coherent text, we must pay attention to each stage of generation, such as content determination, text structuring, aggregation, and generation of anaphoric expression (pronominalization). Among these stages, we are especially interested in the generation of anaphoric expression focusing on zero pronouns, because generating the appropriate zero pronoun is directly connected with text coherence. Consider the following short text.

(1) **Na**-nun (I, topic) cinyel-toyn (on display) **os** (the dresses) cwung (one of) maum-ye tu-nun kes-i iss-ess-ta (attracted me).
(One of the dresses on display attracted me.)

(2) **[Na**-nun (I, topic), ø**]**[1] **[os**-oul (dress, object), ø**]** ip-e po-ass-ta (putted on).
(<u>I</u> (ø) putted <u>it</u> (ø) on.)

(3) Haciman (however), **[na**-nun (I, topic), ø**]** **[os**-i (dress, subject), ø**]** nemu (too) khesu (big) **[os**-ul (it, object), ø**]** sal-swu (buy) eps-ess-ta.(can not).
(However, the <u>dress</u> (ø) was too big so that <u>I</u> (ø) cannot buy <u>it</u> (ø))

In the above text, 'na (I)' appears repeatedly as a topic[2] in sentence (1), (2), and (3), and 'os (dress)'

[1] A bracketed noun, which means the unexpressed argument of the verb, is a *zero pronoun*. Generally, this kind of omitted element caused by the *zero anaphora* phenomenon is called *zero pronoun*, *zero element, zero anaphor*, or *null element*. In this paper, we call the omitted element a *zero pronoun.*

appears repeatedly as an object and a subject in sentence (2) and (3). Korean is a highly context-dependent language, and any arguments recoverable from the context are freely dropped. In the above text, ellipsis of redundant nouns, 'na (I)' and 'os (dress)', in sentence (2) and (3) is recommended to generate a natural Korean text. Otherwise, the text is not coherent because of redundancy.

Our goal is to generate natural anaphoric expressions in Korean, particularly the zero pronoun, using a *Cost-based Centering Model* which considers the inference cost. In this paper, the cost-based centering model refers to the revised centering model by Strube and Hahn (1999), which extends the original 4 transition types to 6 types and defines the cost between transition pairs with respect to the cost for inferring.

This paper is organized as follows. In Section 2, we describe the centering model, which is the main background knowledge for our algorithm to generate anaphoric expression. In Section 3, we briefly describe related works on the generation of anaphoric expressions. In Section 4, we investigate the characteristics of the zero pronoun in Korean, and in Section 5 we describe a cost-based centering model and our proposed algorithm. In Section 6, we discuss the experimental validation. Finally, in Section 7, we summarize the features of our work and future work.

2 The Centering Model

The centering model (Grosz et al., 1986; 1995) provides a framework for the interaction of cohesion and salience in the internal organization of a text. The model is formalized in terms of Cb, the *backward-looking center*, Cf, a list of *forward-looking centers* for each utterance U_n, (i.e., n^{th} utterance or sentence), and Cp, *preferred center* which is the most salient candidate for subsequent utterances. $Cf(U_n)$, i.e., the entities mentioned in U_n are ranked by some measures such as a grammatical role. $Cp(U_n)$ is the highest ranked center of $Cf(U_n)$ and is predicted to be $Cb(U_{n+1})$. If two successive utterances have no reference in common, the second will have no Cb.

Transition type across pairs of adjacent utterances is defined in terms of two factors: cohesion and salience. Cohesion is achieved if the $Cb(U_{n-1})$ and the $Cb(U_n)$ are the same, and salience is achieved if the $Cb(U_n)$ and the $Cp(U_n)$ are the same.

The model consists of three constraints and two rules.

▣ **Constraints**

1. There is precisely one Cb in U_n.
2. Every element of $Cf(U_n)$ must be realized in U_n.
3. $Cb(U_n)$ is the highest-ranked element of $Cf(U_{n-1})$ that is realized in U_n.

▣ **Rules**

1. If some elements of $Cf(U_{n-1})$ are realized as a pronoun in U_n then so is $Cb(U_n)$.
2. Transition types are ordered. *Continue* is preferred over *Retain,* which is preferred over *Smooth-Shift,* which is preferred over *Rough-Shift*.

	$Cb(U_n)=Cb(U_{n-1})$ or undefined $Cb(U_{n-1})$	$Cb(U_n)!=Cb(U_{n-1})$
$Cb(U_n)=Cp(U_n)$	Continue	Smooth-shift
$Cb(U_n)!=Cp(U_n)$	Retain	Rough-shift

Table 1. Transition Types

Although the centering model is attractive to NLP researchers, several issues remain. Several studies have been made on Cf-ranking (e.g., Strube and Hahn, 1999; Turan, 1998; Cote, 1998), because Cf-ranking is language-dependent. Cf-ranking for Korean is different from Cf-ranking for English in that two languages have different features in terms of word-order and functional typology. In this paper, we followed the Cf-ranking proposed by Roh (2003).

topic > subject > *directly-associated-entity (DAE*) > dir-obj > indir-obj > *immediately pre-verbal entity (IPV)*

The topic-first principle is attributed to the topic prominence of Korean.

3 Related Work

Several studies focus on the problem of anaphoric expression generation. The most primitive method

[2] Korean is a topic-prominent language. Topic, an element which is attached topic marker 'un/nun', not only marks the grammatical function of the head noun, but also adds some special meaning to them like "only", "also/too", "even", "in contrast to."

used for early anaphoric generation (e.g., McDonald, 1980; McKeown, 1985) is to use a simple rule: if the current sentence contains the same word mentioned in the previous sentence, use a pronoun to refer to the word. However, this simple rule tends to over-generate pronouns, which causes serious ambiguity.

Recently, some studies attempted to use the centering model for the generation of anaphoric expression. Kibble (2000) used the model to plan coherent texts and to select anaphoric expressions. He considered different strategies for choosing when to use a pronoun, and found the following to be the best: pronominalize the Cb only after a Continue. However, he did not provide experimental results to verify that the method is superior to other strategies.

Mitsuko et al. (2001) adopted the centering model to generate the zero pronoun in Japanese. In English, all arguments of a verb must be expressed in a sentence, and redundant arguments used in previous sentence are usually replaced by pronouns. However, Japanese allows arguments to be freely omitted when they are recoverable from a given context. Korean is quite similar to Japanese from this perspective. Mitsuko argued that all Cb are generated as zero pronouns in either Continue or Smooth-Shift transitions. This can be interpreted as they prefer the zero pronoun when salience rather than cohesion obtains. However, they did not explain the reason why they regarded salience rather than cohesion as an important factor in the zero pronoun.

4 Zero Pronoun in Korean

From the perspective of interpretation of zero pronoun, several studies (e.g., Kim, 1994; Kim, 1999; Ryu, 2000) about zero pronoun were performed by linguists in Korea. Most Korean linguists agree that the zero pronoun generally comes from the continuity of topic, salience of topic, and redundancy of discourse.

More concretely, from the perspective of information structure[3] proposed by Vallduvi (1990), Kim (1999) investigated the conditions of the zero pronoun in Korean: redundant *focus*, redundant *link*, and redundant *tail*. However, these conditions cannot be applied easily, because the focus, link, tail, and their redundancy are not automatically detected, and are determined pragmatically rather than structurally.

Kim (1999) also proposed certain conditions when the zero pronoun would be prohibited, claiming that old information that changes its role is not omitted. For example, when old information in the current sentence becomes a new topic in the next sentence, the role of the old information changes from the old focus to link for the new topic. This situation frequently occurs in the process of topic transition to expand the content of text. From the perspective of Centering, this condition can be interpreted to mean that the Cp of Retain which was one element of the previous sentence cannot be omitted in the transition sequence of Continue, Retain, and Smooth-Shift. Because this transition sequence, called a *Topic-Shift-Sequence* in this paper, is used to smoothly change the current topic to a new topic, and the new topic is generally realized as Cp in Retain. This will be examined as one of the hypotheses to generate anaphoric expression in the next Section.

Ryu (2000) investigated the zero pronoun in terms of Centering. He postulated that the zero pronoun is used to continue the center in Korean, i.e., zero pronoun usually comes from the Cb of Continue. This is supported by many Korean linguists. He also clarified that the zero pronoun rarely appears in written texts when compared with spoken texts.

In Ryu's experimental results, the zero pronoun in Smooth-Shift is worthy of attention. He counted the zero pronouns which comes from the Cb of each transition in four kinds of texts—written, quasi-written, quasi-spoken, and spoken, respectively. In written texts, only 6% of Cb in Smooth-Shift is omitted, and 86% is overtly expressed as a topic with topic marker. Similarly, in quasi-written texts only 12% of Cb in Smooth-Shift is omitted. This phenomenon contrasted with Mitsuko 's generation policy of zero pronoun: generate Cb as a zero pronoun in Smooth-Shift. Perhaps this differ-

[3] In information structure, the sentence is articulated into a trinomial hierarchical structure consisting of 'focus' and 'ground', with the latter further subdivided into 'link' and 'tail'. The focus corresponds to new information unknown to the hearer within a sentence. The ground is the complement of the focus. A link is an address pointer in the sense that it directs the hearer's knowledge-store, which is the information-anchoring role of the ground. The tail is the complement of the link within the ground.

ence is caused by the characteristics of the texts used in the experiments.

To summarize previous research related to zero pronoun in Korean, we conclude that the zero pronoun generally comes from the Cb of Continue. However, some problems still remain from the generation perspective.

- ♦ Which of Cb in Continue is omitted or not, among Cb of Continues?
- ♦ Ellipsis of Cb in the other transitions except for Continue
- ♦ Pronoun generation except for zero pronoun of Cb and Cf

According to Ryu's experimental results, only 26% of Cb in Continues is omitted from written texts. This means that only a partial portion of Cb in Continues becomes a zero pronoun. Recall that all previous research which used the centering model to generate pronouns or zero pronouns follow this principle: pronominalize (or omit in Japanese) Cb in all Continues. Considering Ryu's experimental results and our experimental results (see Section 6 for more details), this traditional strategy causes a serious over-generation of pronouns, including zero pronouns.

Concerning the second problem, almost all previous research considers only pronominalization in Continue, except for Mitsuko (2000) who included Smooth-Shift as well as Continue. However, we confirm that the zero pronoun of Cb in the other transitions also occurs from our corpus test. Therefore, the previous strategy causes the under-generation of pronouns including zero pronouns in the other transitions except for Continue.

Concerning the final problem, in Korean the redundant noun is generally realized as the original one rather than as a pronoun when the zero pronoun is forbidden, unlike English. Consider the following English sentence.

I love my mother and I cannot imagine the world without **her**.

(1) Na-nun (I) wuli (my) emeni-lul (mother) salang-hako (love) **kunye (her)** eps-nun (without) sesang-un (the world) sangsang-hal-su eps-ta (cannot imagine).
(2) Na-nun (I) wuli (my) emeni-lul (mother) salang-hako (love) **emeni (mother)** eps-nun (without) sesang-un (the world) sangsang-hal-su eps-ta (cannot imagine).

In a English-to-Korean translation, sentence (2) which translates 'her' into 'emeni (mother)' is more natural than sentence (1) which translates 'her' into 'kunye (her)'[4]. In this paper, we consider only two types of noun expression, the original noun and the zero pronoun because of these kinds of Korean-dependent characteristics.

5 Generation of Zero Pronoun

5.1 Cost-based Centering Model

We propose a generation algorithm for anaphoric expression in Korean, particularly a zero pronoun, using a cost-based centering model. The model is the revised centering model by Strube and Hahn (1999), which extends the original 4 transition types to 6 types and defines the cost between transition pairs, with respect to the cost for inferring.

	$Cb(U_n)=Cb(U_{n-1})$ or undefined $Cb(U_{n-1})$	$Cb(U_n)!=Cb(U_{n-1})$
$Cb(U_n)=Cp(U_n)$ and $Cb(U_n)=Cp(U_{n-1})$	Cheap-Continue[5] (CC)	Cheap-Smooth-Shift (CSS)
$Cb(U_n)=Cp(U_n)$ and $Cb(U_n)!=Cp(U_{n-1})$	Expensive-Continue (EC)	Expensive-Smooth-Shift (ESS)
$Cb(U_n)!=Cp(U_n)$	Retain (R)	Rough-Shift (RS)

Table 2. Revised Transition Types (Strube, 1999)

Strube and Hahn (1999) argue that a Smooth-Shift which comes from $Cb(U_n) \neq Cp(U_{n-1})$ is less smooth, i.e., the Smooth-Shift requires a high processing cost, because it contradicts the intuition that a Smooth-Shift fulfills the prediction of the Retain. The same applies to a Continue with this characteristic. For this reason, they separated Expensive-Continue and Expensive-Smooth-Shift from Continue and Smooth-Shift in accordance with the equality of $Cb(U_n)$ and $Cp(U_{n-1})$, as shown in Table 2. These postulations coincide with our intuition.

[4] Kunye (her), corresponding to 'her' in English, is the third personal pronoun referring to a woman in Korean.
[5] In this paper, *Continue* and *Smooth-Shift* among revised transition types in Strube's work (1999) are called *Cheap-Continue* and *Cheap-Smooth-Shift* to distinguish from *Expensive-Continue* and *Expensive-Smooth-Shift.*

In this paper, in order to handle the zero pronoun, six transition types which are shown Table 2 and '*resume*' proposed by Knott et al. (2001) were applied. When an utterance mentions an entity not in the immediate previous utterance, but in the previous discourse, a *resume* occurs.

Many researchers working on the centering model agree that considering adjacent transition pairs rather than particular transition provides a more reliable picture about coherence and anaphora resolution (e.g., Grosz et al., 1995; Strube and Hahn, 1999; Kibble and Power, 1999; 2000). More concretely, Strube and Hahn (1999) proposed to classify all the occurrences of transition pairs with respect to the implied inference costs.[6] In this paper, the pair whose cost is cheap is called a *Preferred Transition Pair*, such as (CC,CC), (CC,R), (R,CSS), etc.

Cheap	(CC,CC), (CC,R), (R,CSS), (CSS,CC), (RS,CSS)
Expensive	Other pairs

Table 3. Cost of Transition Pairs

We investigate some transition pairs proposed by Strube and Hahn (1999), and partially adopt them to generate anaphoric expression, as shown in Table 3.

5.2 Generation Algorithm of Zero Pronoun

As the first step in our proposal, we must examine the reason why the revised transition types and preferred transition pairs considering inference cost are appropriate to the generation of the zero pronoun. We argue that the Cb of Cheap-Continue is more redundantly prominent than the Cb of Expensive-Continue. This assumption is reasonable if we consider that Expensive-Continue follows Retain which smoothly changes the topic. The prominence of Cb in Expensive-Continue decreases because of Retain. The following text, extracted from our corpus, is a description of an exhibition 'Cakwi (a kind of Korean traditional farming tools)', and is a good example to illustrate this phenomenon.

(1) Cakwi-nun (Cakwi, topic) wuli-nala-uy (Korean) centhong-cekin (traditional) nong-kikwu-i-ta (farming tool is) (Cakwi is a Korean traditional farming tool.)

(2) Cakwi-uy (Cakwi, adnom) nal-un (edge, topic) celsaknal (celsaknal) ilako-hanta (is called). (Edge of Cakwi is called celsaknal.)

➔ **CP : nal (edge, topic) CB : Cakwi (adnominal), *R***

(3) Cakwi-nun (Cakwi, topic) hyengtay-ka (shape, subject) dokki-wa (axe) pisus-hata (is similar to). (Cakwi is similar to that of an axe.)

➔ **CP : Cakwi (topic) CB : Cakwi (topic), *EC***

(4) Cakwi-nun (Cakwi, topic) khuki-ey ttala (by its size) taycakwi (big-cakwi), socakwi-lo (small-cakwi) nanwin-ta (is categorized). (Cakwi is categorized as big-cakwi and small-cakwi by its size.)

➔ **CP : Cakwi (topic) CB : Cakwi (topic), *CC***

In the above text, the topic smoothly changes from 'Cakwi' to 'nal (edge of Cakwi)' in sentence (2), and Retain occurs. This implies that the topic of the next sentence is 'nal', and it decreases the prominence of Cb, 'Cakwi', in sentence (2). However, in sentence (3), the topic is returned to 'Cakwi' from 'nal', and Expensive-Continue occurs. In sentence (4), 'Cakwi' is maintained as topic, and Cheap-Continue occurs. In this situation, it is natural that the Cb of sentence (3), 'Cakwi', is less prominent than Cb of sentence (4), 'Cakwi', even though both 'Cakwi' are the same as Cb of Continue transitions. If 'Cakwi' in sentence (3) is omitted, the topic (or subject) of 'pisus-hata (is similar to)' can be misinterpreted as 'nal' not 'Cakwi'.

The basic idea of anaphor generation is that the more the noun is redundantly prominent, the more the noun is pronominalized (or omitted).[7] Accordingly, we postulate that the Cb of Expensive-Continue is less elliptical than that of Cheap-Continue. For this reason, we adopt revised transition types considering the inference cost in order to generate the zero pronoun. The case of Smooth-Shift can also be explained in the same manner. For the same reason, it is reasonable that Cb of Continue which follows Continue is more prominent than the Cb of Continue which follows Rough-Shift. For this reason, we adopt the concept of preferred transition pairs.

With these issues in mind, we first construct the following assumptions to generate anaphoric expressions.

[6] Strube and Hahn (1999) argue that, given a sequence of utterances, *inference cost* is needed to understand them. They claim that the inference cost is 'cheap' if successive facts realize preferred transition pairs; otherwise, it is 'expensive'.

[7] Generally, redundantly prominent noun corresponds to Cb within a sentence.

(1) Do generate zero pronoun minimally in written texts.
(2) Do not pronominalize for new information.
(3) Do not make a zero pronoun when it causes ambiguity.
(4) $Cb(U_n)$ is more elliptical than $Cf(U_n)$.

Based on the above assumptions, our algorithm to generate zero pronoun is as follows.

```
(1)If tr(U_n) = CC and cost(U_{n-1},U_n) = cheap then
      realize Cb(U_n) as zero pronoun
(2)Else if tr(U_{n-1}) = CC and tr(U_n) = R and tr(U_{n+1}) =
      CSS (i.e., if three sentences belong to Topic-
      Shift-Sequence) then
        do not realize Cp(U_n), Cb(U_n), and Cb(U_{n+1})
        as zero pronoun
(3)Else if (tr(U_n) = R or tr(U_n) = CSS) and cost(U_{n-1},U_n) = cheap then
        realize Cb(U_n) as zero pronoun
(4)Else
        do not realize Cb(U_n) as zero pronoun
```

$tr(U_n)$: center transition of nth sentence
$cost(U_{n-1},U_n)$: cost of transition pairs between $tr(U_{n-1})$ and $tr(U_n)$

Figure 1. Generation Algorithm of Zero Pronoun

Compared with the traditional anaphor generation strategy related Continue, the rule (1) which adopts an inference cost is more restrictive.

The following example text, which describes an exhibition ‘Paymili (a kind of Korean traditional timber tool)’, is a good example to illustrate rule (2). In this text, the topic changes from ‘Paymili’ to ‘Namaksin (wooden shoes)’ using the Topic-Shift-Sequence. In this process of topic change, Cp of sentence (3) occurring Retain, ‘Namaksin’, should not be omitted in order to imply topic change, and Cb, ‘Paymili’, had better not be omitted in order to smoothly change an old topic (Cb in sentence (3)) to a new topic (Cp in sentence (3)). Similarly, Cb which is equal to Cp in sentence (4) occurring Cheap-Smooth-Shift, ‘Namaksin’, had better not be omitted in order to emphasize a new topic. Therefore, we argue Cp and Cb of Retain and Cb(=Cp) of Cheap-Smooth-Shift as an unadvisable zero pronoun condition under the Topic-Shift-Sequence.

(1) Paymili-nun (Paymili) wuli-nala-uy (Korean) centhong-cekin (traditional) mokcey (timber) yencang-i-ta (tool is). (Paymili is a Korean traditional timber tool.)

(2) [Paymili-nun (topic), **ø]** namaksin-ul (wooden shoes) kkak-ul (cutting) ttay (when) naypu-uy (inside) hyengtay-lul (shape) cap-nuntey (when forming) ssu-in-ta (is used). (Paymili is used for forming the inside shape when cutting wooden shoes.)
➔ CP : Paymili (topic) CB : Paymili (topic), *CC*, cost(1,2) : cheap

(3) Namaksin-un (wooden shoes) **Paymili**-lo (with Paymili) sin-uy (shoe’s) moyang-ul (shape) kolu-ko (raking and), Hopikhal-ul (Hopikhal) iyong-hay (using) kkakk-nun-ta (cutting). (Wooden shoes are made by raking in the shape of a shoe with Paymili and cutting using Hopikhal)
➔ CP : Namaksin (wooden shoes, topic) CB : Paymili (adverb), *R*, cost(2,3) : cheap

(4) Namaksin-un (wooden shoes) pi o-nun (rainy) nal (days) cwulo (usually) sin-ess-nun-tey (are worn and) otong-namwu-na (paulownia tree or) petunamu-lo (willow from) mantul-ess-ta (are made). (Wooden shoes are usually worn on rainy days and are made from the paulownia tree or willow).
➔ CP : Namaksin (wooden shoes, topic) CB : Namaksin (wooden shoes, topic), *CSS*, cost(3,4) : cheap

According to the experimental results of Ryu (2000), the ellipsis ratio of Cb is proportional to the order of Continue, Retain, Smooth-Shift, and Rough-Shift. However, the ellipsis ratio in Retain and Smooth-Shift are low compared with that in Continue, and there is only a slight difference between the ellipsis ratio of Retain and that of Smooth-Shift. Obviously, the ellipsis ratio in Rough-Shift is too low. Therefore, we exclude Rough-shift, and propose rule (3) for Retain and Smooth-Shift under the condition that they do not belong to the Topic-Shift-Sequence. In this algorithm, we do not consider the ellipsis of other $Cf(U_n)$ except for $Cb(U_n)$.

6 Experiments

For an objective evaluation of our proposed algorithm, we investigated the phenomenon of Cb ellipsis from real texts. We collected 87 texts with 15 sentences on average, from three genres, news, story, and descriptive texts. The descriptive texts were gathered from the on-line museum site, ‘the national folk museum of Korea’.[8] We manually recovered the omitted elements of collected texts. In this process, we did not recover the generic pronoun ‘wuli (we)’.

As shown in Table 4, without the inference cost, 175 out of 374 Cb in Continues are realized as zero

[8] Our proposed algorithm will be used to upgrade the XExplainer system (Roh, 2001) which produces a *description* for commodities in Korean. The collected texts are similar to the domain of XExplainer in that they *describe* each exhibition. For this reason, we choose descriptive texts.

pronoun, i.e., 46% of Cb in Continues is omitted. With inference cost, the number, 374, is in turn divided into two classes: 203 Cheap-Continues and 171 Expensive-Continues, and 151 Cb out of 203 Cheap-Continues are omitted. Therefore, 86% (151/175) zero pronouns come from Cb of Cheap-Continues, not of Expensive-Continues. However, by considering 25% ((203-151)/203) of Cb in Cheap-Continues are not omitted, the issue remains concerning which Cb of Cheap-Continue should be omitted or not among the set of Cheap-Continues.

Transition	Without inference cost	With inference cost
CC	175(374)[9], 46%	151(203), 74%
EC		24(171), 14%
R	59(218), 27%	
CSS	34(86), 39%	29(47), 61%
ESS		5(39), 12%
RS	10(104), 9%	

Table 4. Ellipsis of Cb in each Transition

Transition pairs (X : any transition)		Cost of transition pairs	
		Cheap	Expensive
CC	(X, **CC**)	144(172), 83% ➔ **(Rule 1)**	7(31), 22%
R	(X, **R,** ¬CSS)	21(44), 47% ➔ **(Rule 3)**	36(162), 22%
	(CC, **R**, CSS)	2(12), 16% ➔ **(Rule 2)**	
CSS	(¬R, **CSS**)	27(35), 77% ➔ **(Rule 3)**	
	(CC, R, **CSS**)	2(12), 16% ➔ **(Rule 2)**	

Table 5. Ellipsis of Cb in Transition Pairs

The answer can be found in Table 5. Here, 172 out of 203 Cheap-Continues belong to cheap pairs, and the remaining 31 belong to expensive pairs. Moreover, there are 144 ellipses of Cb out of 172 Cheap-Continues associated with cheap pairs, and there are only 7 ellipses of Cb out of 31 Cheap-Continues associated with expensive pairs. To summarize, 82% (144/175) of zero pronouns in Continues come from the Cheap-Continues associated with cheap pairs. Accordingly, we roughly estimate that Cb in Cheap-Continue associated with cheap pairs is realized as a zero pronoun. Although this conclusion causes a slight under-generation of zero pronoun from the viewpoint of our experimental results, this satisfies our first assumption. If we follow the traditional anaphor generation strategy, generating Cb as a zero pronoun in all Continues, 374 zero pronouns occur from Continues. This causes an excessive over-generation of the zero pronoun. However, our algorithm generates only 172 zero pronouns from Continues. Accordingly, we conclude that rule (1) in Figure 1 is a good indicator for the ellipsis of Cb in Continues, and is more restrictive and elaborate than traditional strategies.

Concerning Retain, 27% Cb of total Retains is omitted, and more concretely, 47% Cb of Retains, which are associated with cheap pairs and which do not belong to the Topic-Shift-Sequence, is omitted. 88% Cb of Retains, which are associated with cheap pairs and which belong to the Topic-Shift-Sequence, is not omitted

Concerning Smooth-Shift, without the inference cost, 39% Cb of total Smooth-Shifts is omitted. With inference cost, the number, 86, is in turn divided into two classes: 47 Cheap-Smooth-Shifts and 39 Expensive-Smooth-Shifts. Considering the ellipsis ratio, Cb in Cheap-Smooth-Shift is more elliptical than the Cb in Expensive-Smooth-Shift. More concretely, 77% Cb of Cheap-Smooth-Shifts, which are associated with cheap pairs and which do not belong to the Topic-Shift-Sequence, is omitted. 88% Cb of Cheap-Smooth-Shifts, which are associated with cheap pairs and which belong to the Topic-Shift-Sequence, is not omitted..

Therefore, we estimate that Cb of Retain and Cb of Cheap-Smooth-Shift in the Topic-Shift-Sequence are not realized as zero pronouns, as indicated rule (2) in Figure 1. Additionally, we found that 92% Cp of Retains which belong to Topic-Shift-Sequence is not omitted.

However, rule (3) in Figure 1 is open to discussion because compared with our experimental results, it causes the over-generation of zero pronoun from Cb in Retains. However, in the case of Smooth-Shift, it is effective. Compared with approach of Mitsuko (2001), rule (3), generation of zero pronoun in Smooth-Shift is more elaborate without excessive over-generation.

[9] Parenthesized number means the total number of transitions, and the number in front of the parenthesis means the number of transitions in which Cb is omitted. The percent means the ratio of the two numbers.

7 Conclusion

In this paper, we propose an algorithm for the generation of anaphoric expression, the zero pronoun, in Korean. Our algorithm is based on the cost-based centering model, which extends transition types and defines the cost of transition pairs with respect to the inference cost. Using the model, we resolve both the over-generation of the zero pronoun in Continue and its under-generation in other transitions. We also propose a rule in which ellipsis of Cb or Cp is inadvisable. According to our experimental results, the Cb of cheap transition is more elliptical than that of expensive transition. In addition, the Cb of transition associated with cheap pairs is more elliptical than that of a transition associated with expensive pairs.

This paper did not handle the ellipsis of Cf elements except for Cb. The Pronoun Rule of the centering model applies only to the anaphoric expression which is the Cb of the current sentence. With respect to all other anaphoric expressions except for Cb in the current sentence, the centering model is under-specified. However, there are many omitted elements which are not Cb in our experiment, and they are left as future work. Additionally, the practicality of the proposed method will also be verified through a more reliable evaluation methodology in a real generation system.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc), and also partially by the Brain Korea 21 Project in 2003.

References

Cote, S. 1998. *Ranking Forward-Looking Centers*, Centering Theory in Discourse. Oxford: Clarendon Press, pp55-71

Grosz, B.J. and Sidner, C.L. 1986. *Attention, intentions, and the structure of discourse,* Computational Linguistics, pp175-203

Grosz, B.J., Joshi, A.K. and Weinstein, S. 1995. *Centering: A Framework for Modeling the Local Coherence of Discourse*, Computational Linguistics 21(2), pp203-225

Kibble, R. and Power, R. 1999. *Using centering theory to plan coherent texts*, In Proceedings of the 12th Amsterdam Colloquium.

Kibble, R. and Power, R. 2000. *An integrated framework for text planning and pronominalisation* Proceedings of the 1st International Conference on Natural Language Generation (INLG-2000), Mitzpe Ramon, Israel, pp77-84

Kim, M. K., 1999. *Conditions on Deletion in Korean based on Information Packaging*, Discourse and Cognition 1(2), pp61-88

Kim, M.Y. 1994. *The Centering of Korean discourse*, Ms thesis, Seoul National University

Knott, A., Oberlander, J., O'Donnell, M. and Mellish, C. 2001. *Beyond elaboration: the interaction of relations and focus in coherent text*, In T. Sanders, J. Schilperoord and W. Spooren (eds.) Text representation: linguistic and psycholinguistic aspect. Amsterdam: Benjamins, pp181-196

McDonald, D.D. 1980. *Natural Language Production as a Process of Decision Making under Constraint*, Ph.D. thesis, MIT.

McKeown, K.R. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge, U.K.: Cambridge University Press.

Mitsuko,Yamura-Takei, Fujiwara, M., and Aizawa, T. 2001. *Centering as an Anaphora Generation Algorithm: A Language Learning Aid Perspective*, NLPRS 2001, Tokyo, Japan, pp557-562

Roh, J.E., Kang, S.J. and Lee, J.H. 2001. *Korean Text Generation from Database for Homeshopping Sites*, NLPRS 2001, Tokyo, Japan, pp419-426

Roh, J.E. and Lee, J.H. 2003. *Coherent Text Generation using Entity-based Coherence Measures*, ICCPOL, Shen-Yang, China, pp243-249

Ryu, Byung Ryul, 2001, *Centering and Zero Anaphora in the Korean Discourse,* Seoul National University, Ms Thesis

Strube, M. and Hahn, U. 1999. *Functional Centering: Grounding Referential Coherence in Information Structure* Computational Linguistics 25(3), pp309-344

Turan, Umit D. 1998. *Ranking Forward-Looking Centers in Turkish: Universal and Language Specific Properties,* Centering Theory in Discourse. Oxford: Clarendon Press, pp139-160

Vallduvi, E. 1990. *The Informational component, doctorial dissertation*, University of Pennsylvania

S-clause Segmentation for Efficient Syntactic Analysis Using Decision Trees

Mi-Young Kim **Jong-Hyeok Lee**
Div. of Electrical and Computer Engineering
Pohang University of Science and Technology (POSTECH) and Advanced Information Technology Research Center(AlTrc)
San 31 Hyoja-dong, Nam-gu, Pohang 790-784, R. of Korea
colorful@postech.ac.kr jhlee@postech.ac.kr
fax: +82-54-279-5699 **fax: +82-54-279-5699**

Keywords : S-clause segmentation, Decision trees, Syntactic analysis, Long sentence analysis

Abstract

In dependency parsing of long sentences with fewer subjects than predicates, it is difficult to recognize which predicate governs which subject. To handle such syntactic ambiguity between subjects and predicates, this paper proposes an "S-clause" segmentation method, where an S(ubject)-clause is defined as a group of words containing several predicates and their common subject. We propose an automatic S-clause segmentation method using decision trees. The S-clause information was shown to be very effective in analyzing long sentences, with an improved performance of 5 percent.

1 Introduction

The longer the input sentences, the worse the parsing results are, since problems with syntactic ambiguity increase drastically. In our parser, subject errors form the second largest error portion, as 24.15% of syntactic parsing errors (see Table 1). Although the dependency errors in NP form the largest error portion, these errors are not significant since many applications (e.g. MT systems) using parsers deal with the NP structure as a one unit and do not analyze the syntactic relations within NP. So, this paper proposes a method to resolve subject dependency error problems. To improve the dependency parsing performance, we need to determine the correct dependency relations of subjects.

In most cases, a long sentence has fewer subjects than predicates. The reason is that several predicates can share one subject if they require the same word as their subject, or that the subject of a predicate is often omitted in a Korean sentence. So, in a long sentence, it is difficult to recognize the correct subject of some subjectless VPs. This paper proposes an S(ubject)-clause segmentation method to reduce ambiguity in determining the governor of a subject in dependency parsing. An S(ubject)-clause is defined as a group of words containing several predicates and their common subject. An S-clause includes one subject and several predicates which share the subject. The S-clause segmentation algorithm detects the boundary of predicates which share a common subjective word. We employ the C4.5 decision tree learning algorithm for this task.

The next section presents the background of previous work on sentence segmentation and clause detection. Next, dependency analysis procedure using S-clauses in Korean will be described. Afterwards, the features for decision tree learning to detect S-clauses will be explained, and some experimental results will show that the proposed S-clause segmentation method is effective in dependency parsing. Finally, a conclusion will be given.

dependency tree errors	subject-predicate dependency errors	predicate-predicate dependency errors	adjunct-predicate dependency errors	complement-predicate dependency errors	dependency errors within NP	dependency errors resulting from POS-tag errors
error %	24.15%	14.29%	17.35%	8.84%	27.55%	7.48%

Table 1. Dependency Tree Errors for 10,000 Test Sentences (avg 19.27 Words/sentence)

2 Previous Work

A considerable number of studies have been conducted on the syntactic analysis of long sentences. First, conjunctive structure identification methods have been proposed (Agarwal 1992;Jang 2002;Kurohashi 1994;Yoon 1997). These methods are based on structural parallelism and the lexical similarity of coordinate structures. While they perform well in detecting the boundary of a coordinate structure, they cannot determine the boundary of predicates that share a common subject. In addition, some papers insist that coordinate structure identification is impossible since Korean coordinate sentences do not maintain structural parallelism (Ko 1999).

Second, several studies have been made on clause segmentation (identification, splitting) (Sang and Dejean 2001). The clause seems to be a natural structure above the chunk (Ejerhed 1998). Clause identification splits sentences that center around a verb. The major problem with clause identification concerns the sharing of the same subject by different clauses (Vilson 1998). When a subject is omitted in a clause, Vilson(1998) attached the features of the previous subject to the conjunctions. However, the subject of a clause is not always the nearest subject. Therefore, a new method is required to detect the correct subject of a clause.

In addition, many studies have focused on segmentation in long sentences. Some try to segment a long sentence using patterns and rules and to analyze each segment independently (Doi 1993; Kim 1995; Kim 2002 ;Li 1990). Similarly, an intrasentence segmentation method using machine learning is proposed (Kim 2001). Although this method reduces the complexity of syntactic analysis by segmenting a long sentence, the ambiguity problem with the dependency of subjects remains unsolved. Further, Lyon and Dickerson take advantage of the fact that declarative sentences can almost always be segmented into three concatenated sections (pre-subject, subject, predicate) which can reduce the complexity of parsing English sentences (Lyon and Dickerson 1995; Lyon and Dickerson 1997). This approach is useful for a simple sentence that contains a subject and a predicate. A long sentence generally contains more than a subject and a predicate. Therefore, the segmentation methods proposed by Lyon and Dickerson are inefficient for parsing long sentences. In studies on segmenting long sentences, little attention has been paid to detecting the boundaries of predicates which share a common subject.

To determine the correct subject of some subjectless VPs, we define the 'S-clause' and propose an S-clause segmentation method. In previous work, a clause is defined as a group of words containing a verb, and previous researchers split sentences centering around a verb to detect clauses. By contrast, we split sentences centering around a subject. So we call the proposed segment 'S(ubject)-clause' to distinguish it from a clause.

3 Dependency Analysis in Korean Language

3.1 Dependency Analysis Procedure

This section overviews our dependency analysis procedure for the Korean language.

1. Chunking
2. Detect clauses
   (Using clauses, determine the heads of arguments(except subjects)

3. Detect S-clauses
   (Using S-clauses, determine the heads of subjects and the heads of non-arguments(adjuncts))

First, we determine NP- and VP-chunks following the method of Kim (Kim et al, 2000). Next, we bind a predicate and its arguments to determine the heads of arguments using subcategorization and the selectional restriction information of predicates. This procedure is similar to the clause detection procedure. In this procedure, we also determine the grammatical function of unknown case words according to Lee's method (Lee et al, 2003).

It is important to identify the subject grammatical function of unknown case words correctly, since one S-clause is constructed per subject.

In Korean, arguments of predicates, especially subjects, are often omitted in a sentence. We leave the dependency relations of subjects unconnected, since ambiguity occurs when detecting the heads of subjects.

Third, using predicate information and grammatical function detection results after clause detection, we detect S-clauses. And then, using S-clauses, we determine the dependency relations between subjects and predicates. It can be also helpful in determining the heads of adjuncts, since their heads can be found within an S-clause boundary.

3.2 Dependency Analysis based on S-clauses

Before S-clause segmentation, we have determined the dependency relations between arguments (except subjects) and predicates. Next, we determine the heads of subjects after S-clause segmentation. Although we assume that all the predicates in an S-clause require the subject within the S-clause, some S-clause segmentation errors may exist. To recover the S-clause segmentation errors, we use selectional restriction information to find the relevant head of a subject. We regard the head of the subject within an S-clause as the farthest predicate in the S-clause which requires the concept of the subject.

Still, the dependency relations of adjuncts and those of predicates are not determined. The heads of adjuncts and those of predicates are dependent on the nearest head candidate not giving rise to crossing links. Using S-clauses, we can accomplish dependency parsing simply and effectively.

4 S-clause Segmentation based on Decision Tree Learning

4.1 The C4.5 Learning Algorithm

Decision tree induction algorithms have been successfully applied to NLP problems such as parsing (Magerman 1995;Haruno et al 1998), discourse analysis (Nomoto and Matsumoto 1998), sentence boundary disambiguation (Palmer 1997), phrase break prediction (Kim 2000) and word segmentation (Sornertlamvanich et al 2000). We employed a C4.5 (Quinlan 1993) decision tree induction program as the learning algorithm for S-clause segmentation.

The induction algorithm proceeds by evaluating the information content of a series of attributes and iteratively building a tree from the attribute values, with the leaves of the decision tree representing the values of the goal attributes. At each step of the learning procedure, the evolving tree branches from the attribute that divides the data items with the highest gain in information.

Branches will be added to the tree until the decision tree can classify all items in the training set. To reduce the effects of overfitting, C4.5 prunes the entire decision tree after construction. It recursively examines each subtree to determine whether replacing it with a leaf or branch will reduce the expected error rate. Pruning improves the ability of the decision tree to handle data which is different from the training data.

4.2 Features

This section explains the concrete feature setting we used for learning. The S-clause is a broader concept than the clause. In order to determine the S-clauses, we must choose the clauses that are suitable for addition to the S-clause. Since the head word of a clause is the predicate in the clause, we merely use predicate information. The feature set focuses on the predicates.

An S-clause can be embedded in another S-clause. Therefore, we should learn two methods to detect the left boundary and right boundary of an S-clause independently.

We should include one subject between the left boundary and the right boundary of an S-clause. We call the subject to include in an S-clause the 'target subject'.

First, when we detect the left boundary of an S-clause, we consider the predicates between the

1st Feature	Type of a predicate
2nd Feature	Surface form of the last ending of a predicate
3rd Feature	Comma

Table 2: Linguistic Feature Types Used for Learning

Feature Type	**Values**
1st	adnominal, conjunctive, quotative, nominal, final, null
2nd	ㄴ, ㅁ, 기, 음, ㄴ데, ㄴ즉, ㄴ지, ㄹ지, ㄹ지니, 거나, 게, 고, 나, 는데, 는지, 니, 다가, 도록, 든지, 듯이, 라, 려고, 며, 면, 면서, 므로, 아, 아서, 어, 어도, 어서, 어야, 으나, 으니, 으려고, 으며, 으면, 으면서, 으므로, 자, 지, 지마는, null….
3rd	1, 0, null

Table 3: Values for Each Feature Type

'target subject' and the nearest subject which precedes the 'target subject'.

Each predicate has 3 features, as shown in Table 2. The 1st feature concerns the type of a predicate. Next, the 2nd feature takes the value of the surface form of the last ending of a predicate. Korean is an agglutinative language and the ending of a predicate indicates the connective function with the next VP (e.g. '으므로(because)' indicates it functions as a reason for the next VP).

The 3rd feature deals with the information whether a predicate is followed by a comma or not. The use of a comma to insert a pause in a sentence is an important key to detect an S-clause boundary.

We use 12 features for left boundary detection — 4 predicates, and 3 features for each predicate as summarized in Table 2. The class set consists of 5 values (0~4) to indicate the position of the predicate that becomes a left boundary. If the class value is 0, it means the S-clause includes no predicates preceding the 'target subject'. Other wise, if the class value is 1, it means that that the S-clause includes one nearest predicate which appears preceding the 'target subject'.

The window size of predicates for the left boundary is 4. If there are less than 4 predicates, then we fill the empty features with 'null'.

For right boundary detection, we consider the predicates between the 'target subject' and the next subject following the 'target subject'.

We use 15 features for right boundary detection — 5 predicates, and the same 3 features for each predicate as in Table 2. Among the predicates between 'target subject' and the next subject following the 'target subject', we consider 4 predicates which appear near the 'target subject' and 1 predicate which locates last. The reason that 1 predicate which locates last is considered is as follows: If all the predicates between 'target subject' and the next subject following the 'target subject' require the 'target subject' as their common subject, the right boundary becomes the last predicate among them, since Korean is a head-final language.

Although the feature set is the same as that for right boundary detection, the window size for the right boundary is 5, which is larger than that for the left boundary. The reason is that Korean is a head-final language and the predicates of a subject appear after the subject.

The detailed values of each feature type are summarized in Table 3.

We first detect the S-clause which includes the last subject of an input word set. If an S-clause is detected, we exclude the words which are included in the S-clause from the input word set. Then, we recursively detect the S-clause including the last subject in the remaining word set until there are no subjects in the modified word set.

5 Experimental Results

We evaluated the proposed S-clause segmentation method using the Matec99'[1] test set. We evaluated the following 2 properties of the S-clause segmentation program.

1. The amount of training data vs. S-clause segmentation accuracy vs. parsing accuracy

[1] Morphological Analyzer and Tagger Evaluation Contest in 1999

Number of training sentences	5000	10000	20000	30000	40000	50000
S-clause Precision	82.14%	83.68%	83.70%	84.10%	84.06%	84.40%
S-clause Recall	81.98%	83.54%	83.61%	84.02%	83.98%	84.30%
Parsing Accuracy	86.28%	87.53%	87.86%	88.79%	89.09%	89.12%

Table 4: The Amount of Training Sentences vs. S-clause Accuracy vs. Parsing Accuracy for the 10000 test sentences

	Our parser without S-clause segmentation procedure	Our parser with S-clause segmentation procedure	KN Parser	Korean Yon-sei parser
Accuracy in detecting the head of a subject	51.60 %	84.03 %	74.21 %	Unknown
Parsing Accuracy	84.29%	89.12%	89.93 %	87.30%

Table 5: Parsing Accuracy Comparison (avg 19.27word/sentence)

2. Significance of features

5.1 The Amount of Training Data vs. S-clause Segmentation Accuracy vs. Parsing Accuracy

The test set is different from the training set and the average length of the test sentence is 19.27 words/sentence while that of the training sentence is 14.63 words/sentence. We selected longer sentences as a test set since the S-clause segmentation method is proposed to improve the performance of syntactic analysis in long sentences.

The parsing accuracy is calculated as (correct dependency links)/(all the dependency links). The number of detected dependency links and that of the true dependency links are equal, so parsing accuracy is the same as parsing recall. For the reason, we do not measure the parsing recall separately. However, in the case of S-clauses, the S-clause precision is different from the S-clause recall, since the subject grammatical function detection results for unknown case words are not perfectly correct. We measured the S-clause precision as (correct S-clauses)/(all the detected S-clauses), and the S-clause recall as (correct S-clauses)/(all the true S-clauses).

To show the effectiveness of S-clauses, we compare the parsing result using S-clauses and without S-clauses, and also compare our parser performance with others which analyze similar languages with Korean.

In the experiments, we obtained the following two results.

1. The better the S-clause segmentation performance, the better the parsing accuracy that results.
2. The maximum S-clause accuracy is 84.40% and the maximum parsing accuracy is 89.12% with 50000 training sentences. The test set size is 10,000 sentences.

We will discuss the maximum accuracy of 89.12% compared with the Japanese KN parser, which shows the highest performance in Japanese dependency parsing.

The characteristics of Japanese are similar to the Korean language. So, the mechanism of syntactic analysis in Korean can also be applied to Japanese. In Japanese dependency parsers and Korean dependency parsers, the KN parser shows the highest performance. In addition, we can freely obtain the programs. So, we compare the performance of our parser with that of the KN parser. To do this, we need a bilingual test corpus. We obtain 10,000 Japanese test set by translating the 10,000 Korean test set using Korean-to-Japanese machine translation system of our own. Then, several researchers specializing in Japanese manually corrected the translation results. We experimented on the performance of the KN parser using these 10,000 Japanese sets.

To detect the head of a subject, the KN parser uses only some heuristics (Kurohashi 1994). As shown in Table 5, the performance of our parser

Feature	**Accuracy Change**	**Feature**	**Accuracy Change**
1st type	**-7.34%**	3rd type	-0.04%
1st surface form	-1.15%	3rd surface form	-0.02%
1st comma	-2.42%	3rd comma	-0.82%
2nd type	-0.32%	4th type	-0.0%
2nd surface form	-0.23%	4th surface form	-0.0%
2nd comma	-5.29%	4th comma	-0.01%

Table 6: S-clause Accuracy Change When Each Attribute for Left Boundary Removed

Feature	**Accuracy Change**	**Feature**	**Accuracy Change**
1st type	-3 %	**3rd comma**	**-3 %**
1st Surface form.	-0.8 %	4th type	-0.2 %
1st comma	**-2.7 %**	4th surface form	-0.3 %
2nd type	-0.3 %	4th comma	0.0 %
2nd surface form	-1.3 %	5th type	-0.8 %
2nd comma	**-1.9 %**	5th Surface form.	-0.1 %
3rd type	0.0 %	5th comma	0.0 %
3rd surface form	0.0 %		

Table 7: S-clause Accuracy Change When Each Attribute for Right Boundary Removed

S-clause errors	Subject detection errors	Pos-tag errors	Double subject errors	Left boundary errors	Right boundary errors	Predicate role of adverbials	Other-wise
Error %	25.15%	20.66%	11.38%	16.47%	20.96%	2.00%	3.38%

Table 8: The Type of S-clause Errors

without S-clause segmentation is worse than that of the KN parser. In our parser without S-clause segmentation, a word simply depends on the nearest head not giving rise to crossing links. However, after S-clause segmentation, the performance of our parser is similar to that of the KN parser. The accuracy of our parser in detecting the head of a subject is also better than that of the KN parser.

We also compare the performance of our parser with a Korean Yon-sei dependency parser, as shown in Table 5. The parser using S-clauses outperforms the Yon-sei parser by 1 percent. Since the Yon-sei dependency parser is not an open resource, we simply compare the performance of our parser with that of Yon-sei parser written in Kim (2002). Therefore, the comparison of the performance between our parser and the Korean Yon-sei dependency parser may not be so reasonable.

5. 2 Significance of Features

Next, we will summarize the significance of each feature introduced in Section 4.2. Table 6 and Table 7 illustrate how the S-clause accuracy is reduced when each feature is removed. Table 6 clearly demonstrates that the most significant feature for the left boundary is the type of the previous 1st predicate— we obtain the information from the decision rules that, especially, the 'adnominal' type of the previous 1st predicate is a significant feature. As shown Table 6, 4th predicate information has no effect on the left boundary.

Table 7 demonstrates that the most significant feature for the right boundary is comma information, since the S-clause accuracy without 1st, 2nd or 3rd comma information shows high accuracy decrease. The 5th predicate information is more useful than the 4th predicate. In other words, the last predicate can be the head of a subject than the intermediate predicate.

This result may partially support heuristics; the left boundary would be an adnominal predicate since only adnominal predicates are *followed* by their subjects (other predicates are *preceded* by their subjects). Next, after the comma, a boundary mostly occurs. In particular, we need to concentrate on the types of predicates to attain a higher level of accuracy. To some extent, most features contribute to the parsing performance.

In our experiment, only the surface form of the endings of conjunctive predicates, rather than other predicates, is effective on performance. The reason is that the surface form of the ending of the non-conjunctive predicates does not indicate the connective function with the next VPs.

5.3 Discussion about S-clause Errors

We classify the S-clause errors, as shown in Table 8. Table 8 shows that many S-clause errors are due to the Korean characteristics.

Among the S-clause errors, subject detection errors rank first, which occupy 25.15%. So, the S-clause accuracy result is different from the S-clause recall result.

Next, POS tagging errors result in the S-clause segmentation errors of 20.66 percent.

These two errors occur before S-clause segmentation. So, this is another issue that remains for future work.

Also, double subject errors are 11.38%. Some Korean predicates can require two subjects. This is contrary to our assumption of S-clauses. Since 11.38% is large portion of all the errors, we should consider double subject construction and identify the characteristics of the predicates in double subject constructions.

The right boundary errors are more than left boundary errors. It means that the right boundary detection is more difficult.

Finally, some adverbials, not predicates, can function as predicates of subjects. Since we only detect boundaries focusing on predicates, these adverbials information cannot be used. We should include these adverbials that function as predicates into the S-clause boundary candidates.

6 Conclusion

This paper proposes an S-clause segmentation method to reduce syntactic ambiguity in long sentences. An S(ubject)-clause is defined as a group of words containing several predicates and their common subject. An S-clause includes one subject and several predicates that share the subject.

We have described an S-clause segmentation method that uses decision trees. The experimental results show that the parser using S-clauses outperforms the parser without S-clauses by 5% and also outperforms conventional Korean dependency parsers by 1 percent. To improve the S-clause accuracy, we should detect double subject constructions and adverbials which function as predicates. We plan to continue our research in two directions. First, we will combine our S-clause segmentation method with a coordinate structure detection method and test the parsing results. Second, we will apply it to a machine translation system and translate each S-clause independently and test the translation performance.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center(AITrc) and by the Brain Korea 21 Project in 2003.

References

Rajeev Agarwal, Lois Boggess. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, p.15-21, 1992.

Orasan Constantin. *A hybrid method for clause splitting in unrestricted English texts*. In Proceedings of ACIDCA 2000

Shinchi Doi, Kazunori Muraki, Shinichiro Kamei and Kiyoshi Yamabana. Long sentence analysis by domain-specific pattern grammar, In P*roceedings of the 6th Conference on the European Chapter of the Association of Computational Linguistics*, p.466, 1993.

Eva Ejerhed. Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, p.219-227, 1998.

Masahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. Using Decision Trees to Construct a Practical Parser, In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, p.505-511, 1998.

Jaechol Jang, Euikyu Park, and Dongryeol Ra. A Korean conjunctive structure analysis based on sentence segmentation *(In Korean)*. In *Proceedings of 14th Hangul and Korean Information Processing,* p.139-146, 2002.

Byeongchang Kim and Geunbae Lee. Decision-Tree based Error Correction for Statistical Phrase Break Prediction in Korean. In *Proceedings of the 18th International conference on Computational Linguistics*, p.1051-1055, 2000

Kwangbaek Kim, Euikyu Park, Dongryeol Ra and Juntae Yoon. A method of Korean parsing based on sentence segmentation. In *Proceedings of 14th Hangul and Korean Information Processing,* p.163-168, 2002

Miyoung Kim, Sin-Jae Kang and Jong-Hyeok Lee. *Text Chunking by Rule and Lexical Information.* In proceedings of the 12th Hangul and Korean Information Processing Conference, Chonju, Korea. pp 103~109. 2000

Sungdong Kim and Yungtaek Kim. Sentence analysis using pattern matching in English-Korean machine translation, In *Proceedings of the International Conference on Computer Processing of Oriental Languages*, p.199-206, 1995.

Sungdong Kim, Byungtak Zhang and Yungtaek Kim. Learning-based intrasentence segmentation for efficient translation of long sentences. *Machine Translation* 16:151-174, 2001.

Kwangju Ko. Review of coordinate sentences *(in Korean)*, *Journal of Korean*, 9:39~80, 1999.

Sadao Kurohashi and Makoto Nagao, A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures, *Computational Linguistics*, 20(4):507-534,1994

Vilson. J. Leffa. Clause processing in complex sentences. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, 1998.

Wei-Chuan Li, Tzusheng Pei, Bing-Huang Lee and Chuei-Feng Chiou. Parsing long English sentences with pattern rules. In *Proceedings of the 13th International Conference on Computational Linguistics,* p.410-412, 1990.

YongHun Lee, Mi-Young Kim and Jong-Hyeok Lee. *Grammatical Role Determination of Unknown Cases in Korean Coordinate Structures.* In Proceedings of the 30th Korea Information Science Society Spring Conference, p.543-545, 2003.

Caroline Lyon, and Bob Dickerson. A fast partial parse of natural language sentences using a connectionist method, In *Proceedings of the 7th conference on the European Chapter of the Association of Computational Linguistics*, p.215-222, 1995

Caroline Lyon, and Bob Dickerson. Reducing the complexity of parsing by a method of decomposition. In *Proceedings of the 6th International Workshop on Parsing Technology*, 1997.

David M. Magerman. Statistical Decision-Tree Models for Parsing, In *Proceedings of the 33rd Annual Meeting of Association for Computational Linguistics*, p.276-283. 1995

Tadashi Nomoto and Yuji Matsumoto. Discourse Parsing: A Decision Tree Approach. In proceedings of the 6th Workshop on Very Large Corpora, p.216-224, 1998

David D. Palmer, Marti A. Hearst. Adaptive Multilingual Sentence Boundary Disambiguation, *Computational Linguistics*, 27:241-261, 1997

J. Ross Quinlan. C4.5 Programs for Machine Learning. *Morgan Kaufmann Publishers.* 1993

Erik F. Tjong Kim Sang and Herve Dejean. Introduction to the CoNLL-2001 Shared Task: Clause Identification. In *proceedings of CoNLL-2001*, p. 53-57, 2001.

Virach Sornertlamvanich, Tanapong Potipiti and Thatsanee Charoenporn. Automatic Corpus-Based Thai Word Extraction with the C4.5 Learning Algorithm, In *Proceedings of the 18th International Conference on Computational Linguistics,* p.802-807, 2000

Juntae Yoon, M. Song. Analysis of coordinate conjunctive phrase in Korean *(in Korean), Journal of Korea Information Science Society,* 24:326-336, 1997

Syntactic disambiguation using presupposition resolution

Alistair Knott and Peter Vlugter
Department of Computer Science
University of Otago
alik/pvlugter@cs.otago.ac.nz

Abstract

If a sentence is ambiguous, it often happens that the correct reading is the one which can most easily be incorporated into the discourse context. In this paper we present a simple method for implementing this intuition using the mechanism of presupposition resolution. The basic idea is that we can choose between the alternative readings of an ambiguous sentence by picking the reading which has the greatest number of satistified presuppositions. We present two uses of the disambiguation algorithm in our bilingual human-machine dialogue system.

1 Introduction

Syntactic ambiguity is a well-known problem for natural language interpretation systems, and it is one which becomes increasingly serious as the syntactic coverage of a system increases. A great deal of research in NLP has focussed on how to avoid or alleviate the problem, and several different (and reasonably complementary) approaches have been devised. Four broad classes of approach can be distinguished. The oldest approach involves devising structural heuristics for disambiguation; these are often psycholinguistically-inspired, such as the minimal attachment strategy of Ferreira and Clifton (1986). Another class of approaches rely on the deployment of world or situation knowledge to assess alternative interpretations of a sentence. These approaches require interpretation systems which generate fairly rich semantic representations for sentences, along with large knowledge bases of facts about the domain of interpretation, along with sophisticated inference mechanisms to operate on them. Limitations on the size of knowledge bases and on the speed of theorem-provers mean that this approach is often hard to scale to real applications. (However, there are a number of promising approaches for tackling the scalability problem; see for instance Bos (2001), Hobbs (1993), Stone (1998).) But by far the dominant approach to syntactical ambiguity resolution at the moment involves the use of statistical techniques in one form or another. In particular, using statistical grammars which take into consideration lexical dependencies (see e.g. Collins (1996), Goodman (1998), Magerman (1995)) has proved a very sensitive way of capturing the kind of grammatical contexts that particular words or word combinations typically appear in; this information can be effectively used to decide between alternative parses.

However, there remain some species of ambiguity which are hard to resolve using statistical parsers. It frequently happens that at least two of the alternative readings of a sentence are relatively frequently attested in the corpus used to train the parser. Some of the classical illustrations of syntactic ambiguity can be used to make the point. For instance, there are two readings of the following well-known sentence:

(1) The man saw the girl with the telescope.

In one reading, the PP *with the telescope* attaches to the NP *the girl*, while in the other, it attaches

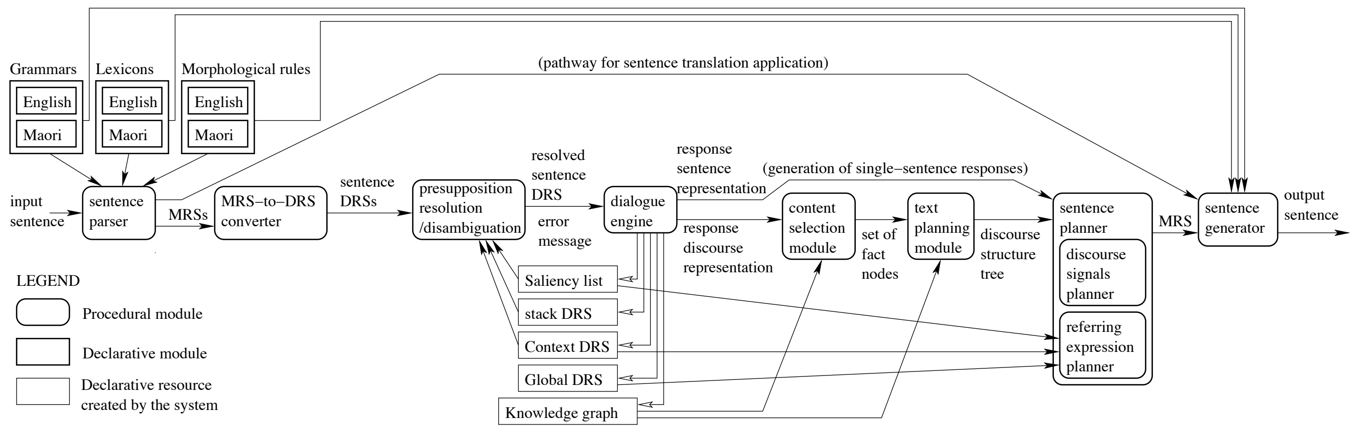


Figure 1: Architecture of the Te Kaitito system

to the VP *saw the girl*. The ambiguity in this sentence is clear, because both the alternative readings describe situations which are consistent with our world knowledge: telescopes can be used for seeing, and people can possess telescopes. In a representative corpus of English, we can expect *with [telescope]* (or perhaps, backing off, *with [instrument]* to occur quite frequently modifying NPs as well as VPs. For the same reason, we cannot expect world knowledge to be very useful in distinguishing between the two alternative parses. A more useful source of information for disambiguation is the immediate discourse context. Simply put, if the reader knows that the man has a telescope *in the current discourse context*, one of the readings is preferred; if the reader knows that the girl has a telescope in the current context, the other reading is preferred.

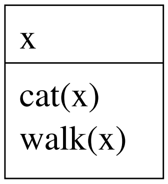
In this paper, we present a simple method for making use of this kind of contextual information in syntactic disambiguation. The key idea is to allow the process of **presupposition resolution** to provide information about the contextual appropriateness of each alternative reading of a sentence (or perhaps each of the readings which a syntactic parser considers reasonably likely). In Section 2, we introduce the general framework we will be working in, by describing our human-machine dialogue system and the semantic representations it uses. In Section 3, we outline the (fairly simple) approach to disambiguation we have in mind, and give an example. In Section 4, we describe one particularly useful application of the approach, for dealing with sentences which have ambiguous information structure.

2 Presuppositional DRT in the Te Kaitito system

Our sentence interpretation system is embedded in a larger human-machine dialogue system called Te Kaitito[1] (see Knott *et al* (2002) for a general overview). The architecture of Te Kaitito is given in Figure 1. When it is the user's turn to contribute to the dialogue, (s)he enters a sentence, in written form. (Te Kaitito is bilingual between English and Māori, so either language can be used.) The sentence is first parsed, using the LKB system (Copestake, 2000), and a set of syntactic analyses are computed. Each analysis is associated with a semantic interpretation in the Minimal Recursion Semantics (MRS) formalism of Copestake *et al*. (1999). Each MRS representation is then converted to a representation in Discourse Representation Theory (DRT) (Kamp and Reyle, 1993), as extended by van der Sandt (1992) to incorporate a treatment of presuppositions. Each of the sentence's DRSs then has its presuppositions resolved using information about the current discourse context, and one of the DRSs is selected as the preferred interpretation of the sentence—this operation is the focus of the current paper. The preferred interpretation is then passed to the dialogue engine, and a response is generated.

[1]Online demos of Te Kaitito can be found at `http://tutoko.otago.ac.nz:8080/teKaitito/`.

Here is a brief introduction to the relevant concepts from DRT. The dialogue context and incoming sentences are modelled with Discourse Representation Structures (DRSs). A DRS is a structure with two fields, one for representing **discourse referents**, and one for representing **conditions** or predications over these referents. DRSs are typically drawn as split boxes, where referents appear at the top, and conditions below. For example, here is the DRS for the sentence *A cat walked*:



The discourse referent x is created by the indefinite NP *a cat*. This shows that *a cat* has introduced a new discourse referent. The conditions $cat(x)$ and $walk(x)$ were placed in the bottom part by the NP *a cat* and the VP *walked*.

A sentence's **presuppositions** are elements of its content which the speaker assumes are already part of the common ground; they are constraints on the kinds of context in which the sentence can be uttered. Here are two examples.

(2) The dog chased a cat.

(3) John's cat slept.

Sentence 2 presupposes that there is a dog in the discourse context (or more precisely, that there is exactly one *salient* dog in the context). Sentence 3 presupposes that there is someone called John, and also that this person has a cat. Presuppositions are triggered by lexical items such as the definite article, proper names, and possessive forms. These triggers determine what is asserted information, and what is presupposed in a given sentence.

As already mentioned, we use a DRT-based treatment of presuppositions as proposed by van der Sandt (1992). A sentence is modelled as an **assertion DRS** and a set of **presupposition DRSs**. The DRSs for Examples 2 and 3 are shown in Figures 2 and 3. Notice that the presupposition DRSs are distinguished by dashed lines.[2]

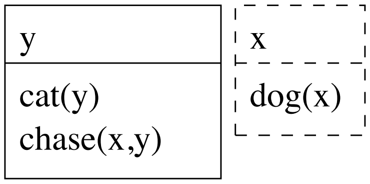


Figure 2: *The dog chased a cat*

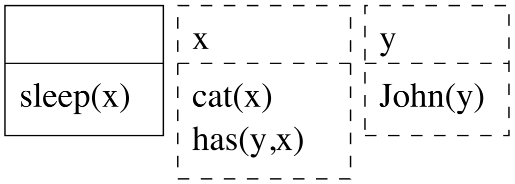


Figure 3: *John's cat slept*

The presuppositions of a sentence need to be **resolved** or **satisfied** in the current discourse context before its assertional content can be processed. In van der Sandt's DRS-based treatment, this is modelled as a binding operation: the referents in each of the sentence's presupposition DRSs need to be bound to referents in the context DRS which have the properties identified in the presupposition DRS. Once this binding has been done, if the presuppositions of a sentence are not satisfied, referents with suitable properties can be (charitably) assumed to exist, and added to the context DRS, in an operation called **accommodation**.

3 Presuppositional weight for sentence disambiguation

Our basic idea for disambiguating between alternative readings of an ambiguous sentence is to choose the reading that can most easily be incorporated into the discourse context. To implement this we use the following general strategy: first, generate the DRS for each alternative interpretation of an ambiguous sentence; then perform presupposition resolution for each interpretation within the current discourse context; finally, choose the most contextually appropriate interpretation. We use the following two principles to determine preference between the alterna-

[2]One difference between our model and that of van der Sandt's is that there is no hierarchy of presuppositions in our model: they all appear at the same level. In van der Sandt's model, a presupposition DRS can itself have presuppositions.

tives:

1. **Accommodation principle:** the most contextually appropriate interpretations of a sentence are those whose presuppositions can be resolved with the minimum amount of accommodated material.

2. **Presuppositional weight principle**: if two interpretations of a sentence can both have their presuppositions resolved *without* accommodation, the most contextually appropriate interpretation is the one with the greatest amount of presuppositional material.

One point to clarify in relation to these principles is how to determine the 'amount' of accommodated or presupposed material in the interpretation of a sentence. We will assume that this relates simply to a count of DRS conditions. For instance, for the accommodation principle, an interpretation which requires only one DRS condition to be accommodated into the discourse context is preferable over an interpretation which requires two conditions to be accommodated. For the presuppositional weight principle, given two interpretations whose presuppositions are both satisfied without recourse to accommodation, we simply count the total number of presupposed DRS conditions in each interpretation and choose the one with the highest total. It may be that there are alternative definitions for this notion of 'amount of semantic material', for instance including a count of the number of referents introduced, or the number of separate presupposition DRSs. But a simple account will suffice for the examples we deal with in this paper.

An example

Let us return to Example (1), repeated below:

(4) The man saw the girl with the telescope.

There are two possible readings for this sentence. Both readings presuppose a man, a girl, and a telescope, and both assert that the man saw the girl. The DRS that represents the reading in which the man used the telescope to see the girl is given in Figure 4. The other possible reading, in which the girl possesses the telescope, is given in Figure 5. Notice that the second reading has more presuppositional content, or greater presuppositional weight, than the first.

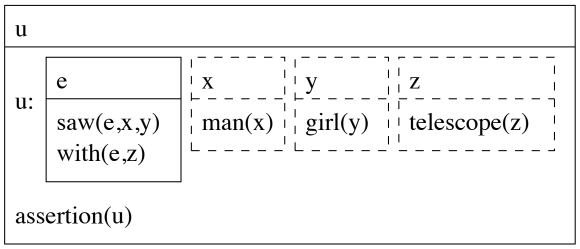


Figure 4: *...[saw [the girl] [with the telescope]]*

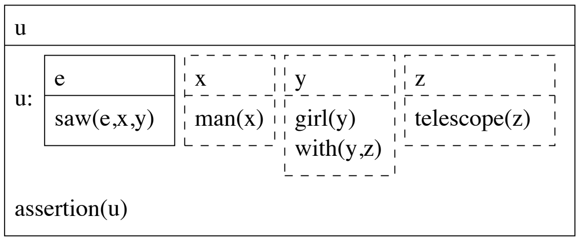


Figure 5: *...[saw [the girl [with the telescope]]]*

If we know of a girl who has a telescope (see e.g. Figure 6(a)), then the reading in which the girl possesses the telescope would be preferred. In a context like this all the presuppositions of both readings could be resolved. However, we can choose the second reading, by the presuppositional weight principle, since it is 'heavier' in resolvable presuppositions. In a discourse context where there is a man, a girl, and a telescope, but the girl does not possess the telescope (see e.g. Figure 6(b)), only the presuppositions of the first reading can be satisified. In this case, this reading is preferred, by the accommodation principle.

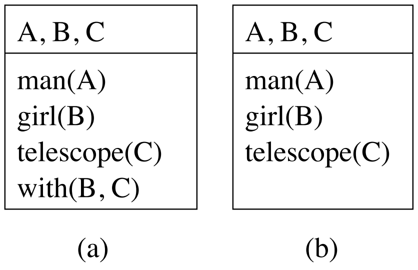


Figure 6: Two alternative contexts for Sentence 1

4 Application: sentences with ambiguous information structure

In this section, we describe a use for the two principles just outlined in the interpretation of sentences with ambiguous information structure. To begin with, we will illustrate this kind of ambiguity. Consider the following sentence:

(5) The cat chased the dog.

The most obvious reading of the sentence is that the speaker is simply asserting a new fact a propos of nothing, namely that the cat chased the dog. However, there are alternative possible readings in which the speaker is answering a wide variety of *questions*: for instance, ‘What did the cat chase?’, ‘What chased the dog?’, ‘What did the cat do to the dog?’ and so on. If the answers were spoken, they would have different prosodic and intonational structures: *THE CAT chased the dog*, *The cat chased THE DOG*, *The cat CHASED the dog*, and so on. In a grammar of English, it is important to distinguish these alternative readings even when there are no prosodic cues; any declarative sentence can genuinely be interpreted in these different ways, and the grammar should reflect this. However, computational grammars typically do not pay much attention to information structure ambiguity,[3] principally because once the ambiguity is created, it is hard to resolve. Information structure ambiguity is prototypically a kind of ambiguity which cannot be resolved using statistical techniques or knowledge about the domain; it requires an analysis of the relation between the sentence and its immediate discourse context.

In this section, we describe how information structure ambiguity of this kind can be resolved using a presuppositional framework. We begin by providing some background about the way questions and answers are represented using presuppositions in our system.

4.1 Background: presuppositions in questions and answers

In the Te Kaitito system we model questions and answers using presuppositions (for details, see de Jager *et al*. (2002)). A question can be thought of as asserting that something is ‘unknown’,[4] and presupposing the content of the query to exist within the current discourse context. For example, consider the question:

(6) What chased the dog?

This question presupposes that something did chase the dog. The only information it asserts is that the something which chased the dog is unknown to the speaker. This is represented as a DRS in Figure 7.

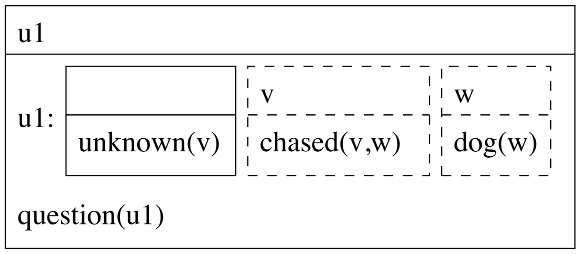


Figure 7: *What chased the dog?*

An answer to a question also makes use of presuppositions. An answer presupposes a question containing an unknown element, and asserts that this unknown element is identical to some referent within the discourse context. An answer to the question in Example 6 could be given as *It was the cat* or just *The cat*. This answer is represented as a DRS in Figure 8. For further details

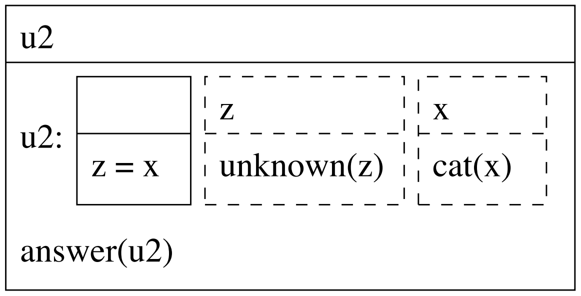


Figure 8: *The cat / It was the cat*

about this presuppositional treatment of questions

[3]Categorial grammar is a notable exception; see Steedman (2000).

[4]Strictly speaking, we are here asserting something about the speaker’s *knowledge* about the object, rather than about the object itself. But in our system, we draw no distinction between this kind of epistemic predicate and ordinary first-order predicates.

and answers, and a discussion of some of its additional benefits in a dialogue management system, see de Jager *et al*. (2002).

4.2 Using presuppositional weight for disambiguation

One could also answer the question in Example 6 using the full sentence *The cat chased the dog* (c.f. Example 5), with understood emphasis *THE CAT chased the dog*. How do we distinguish this reading from the other possible readings? The DRSs in Figures 9-11 show three ways of interpreting *The cat chased the dog*. Using this rep-

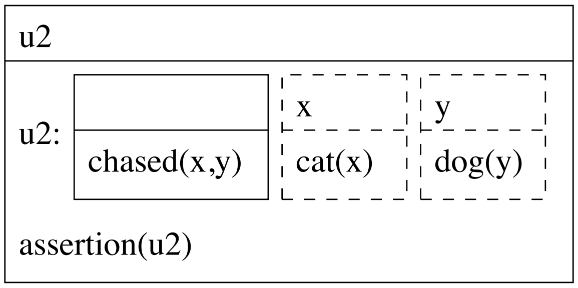


Figure 9: *The cat chased the dog*

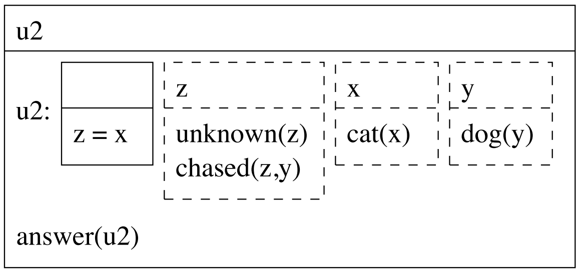


Figure 10: *THE CAT chased the dog*

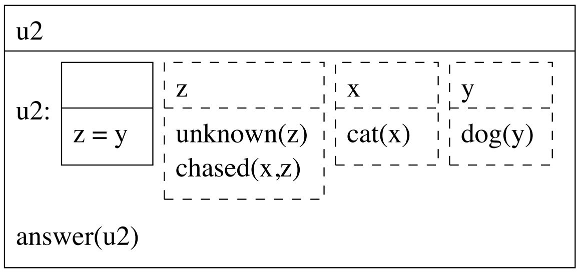


Figure 11: *The cat chased THE DOG*

resentation, the principles presented in Section 3 can be used directly to derive the right interpretation of the sentence, namely that given in Figure 10. Firstly, note that both the interpretation in Figure 9 and that in Figure 10 can have all their presuppositions resolved without accommodation in the context created by the question (Figure 7), while the interpretation given in Figure 11 contains a presupposition which needs to be accommodated in this context (an unknown object which is chased). By the accommodation principle, we can reject the interpretation in Figure 11. Now note that the interpretation in Figure 10 has a larger number of presupposed DRS conditions than the interpretation in Figure 9 (4 versus 2). By the presuppositional weight principle, we can therefore choose the interpretation in Figure 10, which is the correct interpretation for the sentence.

5 Summary and conclusions

In this paper, we have presented a simple approach to sentence disambiguation, which makes use of presupposition resolution, and we have described two example contexts where the approach is useful. We have argued that this approach is likely to be helpful in cases where other methods for disambiguation have difficulties, in particular in cases where statistical methods cannot deliver a clear verdict because two (or more) interpretations of a sentence contain constructions which are frequently attested in training corpora, and where methods based on determining consistency with world or domain knowledge have a similar problem; information structure ambiguity is a good case in point.

Naturally, the information about contextual appropriateness delivered by the presupposition resolution mechanism should just be considered as one source of information about the appropriate reading of a sentence. Other sources of information should also be taken into account. A statistical parser is still likely to be of considerable use in weeding out very unsuitable readings at the very start, because presupposition resolution is computationally quite expensive. And information from world or domain knowledge about the absolute likelihood of alternative readings is also bound to be important. Our main conclusion is simply that information about presuppositions should have a useful role to play in a complete framework for sentence disambiguation.

Acknowledgements

This work was supported by University of Otago Research Grant MFHB10, and by the NZ Foundation for Research in Science & Technology (FRST) grant UOOX02.

References

J Bos. 2001. DORIS 2001: Underspecification, resolution and inference for discourse representation structures. In *Proceedings of ICoS-3: Inference in Computational Semantics*.

M Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Meeting of the ACL*, Santa Cruz.

A Copestake, D Flickinger, I Sag, and C Pollard. 1999. Minimal Recursion Semantics: An introduction. Manuscript, CSLI, Stanford University.

A Copestake. 2000. The (new) LKB system. CSLI, Stanford University.

S de Jager, A Knott, and I Bayard. 2002. A DRT-based framework for presuppositions in dialogue management. In *Proceedings of the 6th workshop on the semantics and pragmatics of dialogue (EDILOG 2002)*, Edinburgh.

F Ferreira and C Clifton. 1986. The independence of syntactic processing. *Journal of Memory and Language*, 25:348–368.

Joshua Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University.

J Hobbs, M Stickel, D Appelt, and P Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63.

H Kamp and U Reyle. 1993. *From discourse to logic*. Kluwer Academic Publishers, Dordrecht.

A Knott, I Bayard, S de Jager, and N Wright. 2002. An architecture for bilingual and bidirectional nlp. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop (ANLP 2002)*.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics. Cambridge, MA, 26–30 Jun 1995*.

M Steedman. 2000. *The syntactic process*. MIT Press/Bradford Books.

M Stone. 1998. *Modality in dialogue: planning, pragmatics and computation*. Ph.D. thesis, Institute for Research in Cognitive Science, University of Pennsylvania.

R Van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377.

The Ins and Outs of Dutch Noun Countability Classification

Timothy Baldwin
CSLI
Stanford University
Stanford, CA 94305 USA
tbaldwin@csli.stanford.edu

Leonoor van der Beek
University of Groningen
Pb 716, 9700 AS, Groningen
The Netherlands
vdbeek@let.rug.nl

Abstract

This paper presents a range of methods for classifying Dutch noun countability based on either Dutch or English data. The classification is founded on translational equivalences and the corpus analysis of linguistic features which correlate with particular countability classes. We show that crosslingual classification on the basis of word-to-word or feature-to-feature mappings between English and Dutch performs at least as well as in-language classification based on gold-standard Dutch countability data.

1 Introduction

The performance of supervised learning methods is conditioned on the quality of annotation and also volume of training data (Hastie et al., 2001). This effect is felt particularly keenly in tasks of high feature dimensionality or low feature–class correlation. In many cases, high-quality data is not available in large quantities, but a large volume of lower-quality data can be accessed (Mitchell, 1999; Banko and Brill, 2001). Alternatively, high-quality data may exist for some parallel task which can be adapted to the task at hand through a lossy feature mapping. This strategy has been adopted successfully in NLP applications such as part-of-speech tagging involving languages with a relative paucity of language resources or annotated data (Yarowsky et al., 2001; Cucerzan and Yarowsky, 2002).

This paper takes a supervised learning task and contrasts the use of a restricted volume of in-language training data with the use of a larger volume of out-of-language training data adapted to the task through a lossy mapping. Our aim in this is to determine the most effective fast-track solution when faced with a novel task in a given language for which high-quality annotated data exists in a closely-related language.

We illustrate this issue by way of a type-level noun countability classification task in Dutch for which we have moderate amounts of high-quality annotated data in English and large amounts of medium-quality annotated data in Dutch (see §2.4). For English, previous research has shown that corpus evidence can be applied successfully to classify unannotated noun types for countability (Baldwin and Bond, 2003a; Baldwin and Bond, 2003b). We extend this research to Dutch and address the question of which of high-quality out-of-language English data and lower-quality in-language Dutch data produces the best Dutch countability classification results, realising that the feature mapping from English-to-Dutch in the first case will necessarily be lossy.

We treat **noun countability** as a lexical property that determines determiner co-occurrence, the ability to pluralise, and enumeration effects. Each Dutch noun type is classified as being countable and/or uncountable, noting that different senses/usages of a given word can occur with different countabilities, cf. *I want a <u>rabbit</u>* $\rightleftharpoons$ *Ik will een <u>konijn</u>* [countable] vs. *I would like some more <u>rabbit</u>, please* $\rightleftharpoons$ *I zou graag nog wat <u>konijn</u> willen* [uncountable]. Knowledge of countability is important both for analysis and generation. In analysis it helps to constrain the set of possible parses and their interpretation. In generation, countability information determines whether a noun can be pluralised and what determiners it can combine with.

The assumption underlying the crosslingual countability classification task is that Dutch and English are sufficiently close linguistically that there is a strong correlation between noun countability in the two languages. Both languages distinguish countable, uncountable and plural only nouns.[1] Although mismatches exist—e.g. *brain* [countable] vs. *hersenen* [plural only], *thun-*

[1] A fourth class of bipartite nouns (e.g. *scissors, trousers*) is generally recognised for English, but has no Dutch correlate.

derstorm [countable] vs. *onweer* [uncountable]—many Dutch words are in the same countability class as their English equivalents (e.g. *car* ⇌ *auto* [countable], *food* ⇌ *eten* [uncountable] and *goods* ⇌ *goederen* [plural only]). One obvious approach, therefore, is to simply map the countabilities of English nouns onto their Dutch counterparts.

A less direct approach to crosslingual countability transfer is to base classification on corpus occurrence with linguistic predictors of the different countability classes, in the manner of Baldwin and Bond (2003a). Linguistic features that are associated with the countability classes often have direct translations in the other language (e.g. syntactic number, co-occurrence with denumerators) or can be mapped onto an equivalent feature (e.g. the English N_1 *of* N_2 construction and Dutch measure noun construction—see §2.2). In some cases however, the mapping is imperfect (e.g. *much* occurs only with uncountable nouns, but the Dutch translation *veel* is also the translation of *many*, and occurs with both uncountable singular and countable plural nouns) or no equivalent exists in one of the languages (e.g. the occurrence of a plural noun as a modifier is a weak indicator of plural only in English, but not in Dutch).

The remainder of this paper is structured as follows. §2 describes the countability classes, the nature and extraction of the features used in the corpus-based method, the feature abstraction method and the gold-standard data. §3 outlines the various classifiers tested in this research. §4 presents and discusses the experimental results. The conclusions of the paper are given in §5.

2 Preliminaries

2.1 Countability classes

Dutch and English nouns are generally considered to belong to one or more of three possible countability classes: countable, uncountable and plural only. **Countable** nouns can be modified by denumerators, prototypically numbers, and have a morphologically marked plural form: *one dog* ⇌ *een hond*, *two dogs* ⇌ *twee honden*. **Uncountable** nouns cannot be modified by denumerators, but can be modified by unspecific quantifiers such as *much* ⇌ *veel*, and do not show any number distinction (prototypically being singular): **one rice* ⇌ **een rijst*, *some rice* ⇌ *een beetje rijst*, **two rices* ⇌ **twee rijsten*. This class includes many abstract nouns, material-denoting nouns, generics and deverbalised nouns. **Plural only** nouns only have a plural form, such as *goods* ⇌ *goederen* and cannot be denumerated. The plural only class is considered to be a closed class in Dutch, and is thus ignored in the classification experiments below.[2] Note that countability distinctions are in fact not categorical (Allan, 1980): prototypical countable nouns can be used in uncountable contexts, forcing a 'substance' interpretation (the **universal grinder**, e.g. *there was deer all over the road* ⇌ *over de hele straat lag hert*) and uncountable nouns can in certain contexts be denumerated, resulting in a 'type' interpretation (the **universal packager**, e.g. *this shop sells three different wines* ⇌ *deze winkel verkoopt drie verschillende wijnen*). However, nouns are generally considered to have a basic classification as countable and/or uncountable.

2.2 Feature space

The feature space used in this research is made up of **feature clusters**, each of which is conditioned on the occurrence of a **target noun** in a given construction. Feature clusters are either one-dimensional (describe a single multivariate feature) or two-dimensional (describe the interaction between two multivariate features), with each dimension describing a lexical or syntactic property of the construction in question. Below, we provide a basic description of the 9 feature clusters used in this research and their dimensionality ($^{[x]}\mathbf{L}$=1-dimensional feature cluster with x unit features for language $\mathbf{L}$, $^{[x \times y]}\mathbf{L}$= 2-dimensional feature cluster with $x \times y$ unit features for language $\mathbf{L}$). For further details and predicted correlations between feature values and particular countability classes for English, the reader is referred to Baldwin and Bond (2003a).

Head noun number:$^{[2]}\mathbf{E} \rightleftharpoons {}^{[2]}\mathbf{D}$ the number of the target noun when it heads an NP

Subject–verb agreement:$^{[2\times2]}\mathbf{E} \rightleftharpoons {}^{[2\times2]}\mathbf{D}$ the number of the target noun in a subject position vs. number agreement on the governing verb

Coordinate noun number:$^{[2\times2]}\mathbf{E} \rightleftharpoons {}^{[2\times2]}\mathbf{D}$ the number of the target noun vs. the number of the head nouns of conjuncts

$\mathbf{N_1}$ *of* $\mathbf{N_2}$/measure noun constructions: $^{[11\times2]}\mathbf{E} \rightleftharpoons {}^{[11\times2]}\mathbf{D}$ the type of the N_1 vs.

[2]But see van der Beek and Baldwin (2003) for classification results over the plural only class.

the number of the target noun (N_2) in an English N_1 *of* N_2 construction or Dutch measure noun construction. N_1 types include COLLECTIVE (*a group of people* ⇌ *een groep mensen*), UNIT (*a kilo of sugar* ⇌ *een kilo suiker*) and TEMPORAL (*a minute of silence* ⇌ *een minuut stilte*).

Occurrence in PPs:$^{[52\times2]_\mathbf{E} \rightleftharpoons [84\times2]_\mathbf{D}}$ the preposition type and presence or absence of a determiner when the target noun occurs in **singular** form in a PP.

Pronoun co-occurrence:$^{[12\times2]_\mathbf{E} \rightleftharpoons [7\times2]_\mathbf{D}}$ what personal, reflexive and possessive pronouns occur in the same sentence as singular and plural instances of the target noun.

Singular determiners:$^{[10]_\mathbf{E} \rightleftharpoons [10]_\mathbf{D}}$ what singular-selecting determiners (e.g. *much*) occur in NPs headed by the target noun in **singular** form.

Plural determiners:$^{[12]_\mathbf{E} \rightleftharpoons [13]_\mathbf{D}}$ what plural-selecting determiners (e.g. *many*) occur in NPs headed by the target noun in **plural** form.

Number-neutral determiners:$^{[11\times2]_\mathbf{E} \rightleftharpoons [13\times2]_\mathbf{D}}$ what number-neutral determiners (e.g. *less*) occur in NPs headed by the target noun, and what is the number of the target noun for each.

The Dutch and English feature clusters represent the same linguistic structures, even if the individual features are not direct translations of each other. The only exception is the N_1 *of* N_2/measure noun construction where markedly different constructions in the two languages express the same concept (a quantity of something) and bring about the same restrictions with respect to countability.

2.3 Feature extraction

We use a variety of pre-processors to map the raw data onto the types of constructions targeted in the feature clusters, namely a POS tagger and a full-text chunker for both English and Dutch, and additionally a dependency parser for English. For Dutch, POS tags, lemmata and chunk data were extracted from automatically generated, fully parsed **Alpino** output (Bouma et al., 2000). For English, we used a custom-built fnTBL-based tagger (Ngai and Florian, 2001) with the Penn tagset, morph (Minnen et al., 2001) as our lemmatiser, an fnTBL-based chunker which runs over the output of the tagger, and RASP (Briscoe and Carroll, 2002) as the dependency parser.

These data sets are then used independently to test the efficacy of the different systems at capturing features used in the classification process, or in tandem to consolidate the strengths of the individual methods and reduce system-specific idiosyncrasies in the feature values. When combining Dutch and English in classification, we invariably combine like systems (e.g. Dutch POS data with English POS data).

The English data was extracted from the written component of the British National Corpus (90m words: Burnard (2000)), and the Dutch data from the newspaper component of the Twente Nieuws Corpus (20m words).[3]

After generating the different feature vectors for each noun based on the above configurations, we filtered out all nouns which did not occur at least 10 times in NP head position according to the output of all pre-processors. This resulted in 20,530 English nouns and 12,734 Dutch nouns.

2.4 Gold standard data

Information about English noun countability was obtained from two sources: **COMLEX** 3.0 (Grishman et al., 1998) and the common noun part of **ALT-J/E**'s Japanese-to-English semantic transfer dictionary. These two resources were combined in two ways: (1) by taking the intersection of positive and negative exemplars for each countability class (the **binary** datasets); and (2) by taking the union of all countabilities for a given word in the two resources and representing it as a single multiclass (i.e. countable, uncountable or countable+uncountable: the **multiclass** dataset). In each case, the total number of training instances is around 6,000 words. To determine the quality of annotation, we hand-annotated 100 unseen nouns and measured the agreement[4] with the gold-standard datasets. The agreement for the binary dataset was 92.4%, and that for the multiclass dataset was 89.8%.[5]

In Dutch, there are two electronic dictionaries with countability information: **CELEX** (Baayen et al., 1993) and the **Alpino** lexicon (Bouma et

[3]`http://wwwhome.cs.utwente.nl/~druid/TwNC/TwNC-main.html`

[4]I.e. the proportion of word-level countability class assignments over which the two systems agreed.

[5]The disparity here is due to the fact that the binary dataset is constructed more conservatively and does not contain any words where there is disagreement in countability between **COMLEX** and **ALT-J/E**.

al., 2000). The latter includes the former as well as the Parole lexicon (no countability information), and has been manually modified and extended. We thus used the **Alpino** data to generate a total of three sets of training data for the monolingual Dutch classifiers in the same manner as for English: separate binary datasets for each of the countable and uncountable classes, and a combined multiclass-based dataset. The total number of training instances in each case is around 14,500, over twice the size of the English datasets.

In order to both evaluate the various classifiers and gauge the reliability of the **Alpino** countability judgements, we manually annotated 196 unseen Dutch nouns, basing judgements on actual usage in the Twente Nieuws Corpus. The agreement in countability judgements between the **Alpino** lexicon and hand-annotated data is 81.1%. This is markedly lower than the agreement for the English datasets, and supports our claims about the relatively low quality of the Dutch **Alpino** data as compared to the English data.

3 Classifier design

We propose a variety of both monolingual (Dutch-to–Dutch = **NN** and English-to-English = **EE**) and crosslingual (English-to-Dutch = **EN**) unsupervised and supervised classifier architectures for the task of learning countability. We employ two basic classifier architectures: (1) a separate binary classifier for each countability class (**BIN**), and (2) a single multiclass classifier (**MULTI**). In all cases, our supervised classifiers are built using TiMBL version 4.2 (Daelemans et al., 2002), a memory-based classification system based on the k-nearest neighbour algorithm.

3.1 Monolingual classifiers

Evidence-based classifiers: $\mathbf{NN}_{\text{BIN}}$(evidence,$*$)

In an attempt to derive a baseline for each countability class/pre-processor system combination, we built a (binary) monolingual unsupervised classifier based on diagnostic evidence. For each target noun, the unsupervised classifier simply checks for the existence of diagnostic data in the output of each of the POS tagger and chunker for the given countability class (**NN(evidence,POS)** and **NN(evidence,chunk)**, respectively). Diagnostic data takes the form of unit features which are uniquely associated with a given countability class, e.g. the determiner *a* $\rightleftharpoons$ *een* co-occurring with a given (singular) noun is a strong indicator of that noun being countable. We perform basic system combination by positively classifying any noun for which either of the two pre-processors produces diagnostic data for the given countability class (**NN(evidence,all)**).

Distribution-based classifiers: $\mathbf{NN}_{\text{BIN}}(\mathbf{feat}_{\text{ALL}})$

Despite our reservations about the quality of countability annotation in the **Alpino** lexicon, we implemented a conventional monolingual classifier based on the full feature set given above (§2.2). In this, we take each target noun in turn and compare its amalgamated value for each unit feature with: (a) the values for other target nouns, and (b) the value of other unit features within that same feature cluster (Baldwin and Bond, 2003b).

In the case of a one-dimensional feature cluster, each unit feature f_s for target noun w is translated into 3 separate feature values:

$$corpfreq(f_s, w) = \frac{freq(f_s|w)}{freq(*)} \quad (1)$$

$$wordfreq(f_s, w) = \frac{freq(f_s|w)}{freq(w)} \quad (2)$$

$$featfreq(f_s, w) = \frac{freq(f_s|w)}{\sum_i freq(f_i|w)}\rangle \quad (3)$$

where $freq(*)$ is the frequency of all words in the corpus. That is, for each unit feature we capture the relative corpus frequency, frequency relative to the target word frequency, and frequency relative to other features in the same feature cluster. Thus, for an n-valued one-dimensional feature cluster, we generate $3n$ independent feature values.

In addition to mapping individual unit features onto triples, we introduce a triple for each feature cluster representing the sum over all member values.

In the case of a two-dimensional feature matrix (e.g. subject-position noun number vs. verb number agreement), each unit feature $f_{s,t}$ for target noun w is translated into $corpfreq(f_{s,t}, w)$, $wordfreq(f_{s,t}, w)$ and $featfreq(f_{s,t}, w)$ as above, and 2 additional feature values:

$$featdimfreq_1(f_{s,t}, w) = \frac{freq(f_{s,t}|w)}{\sum_i freq(f_{i,t}|w)} \quad (4)$$

$$featdimfreq_2(f_{s,t}, w) = \frac{freq(f_{s,t}|w)}{\sum_j freq(f_{s,j}|w)} \quad (5)$$

which represent the $featfreq$ values calculated along each of the two feature dimensions. As for one-dimensional feature clusters, we introduce amalgamated features for each row ($f_{i,*}$) and column ($f_{*,j}$) of the feature matrix, and describe each in the form of 3 values. For further details, see the description of the monolingual English task in Baldwin and Bond (2003a). This abstraction generates a total of 1,664 individual feature values for Dutch.

We learned individual countable and uncountable classifiers from the binary **Alpino** data, averaging the feature values across those from the tagger and chunker in each case.[6]

3.2 Crosslingual classifiers

Translation-based classifier: EN$_{\text{BIN}}$(translate)

Translation-based classification applies the observation that Dutch nouns often take the same countability as their English translation equivalents. First, we derive English countabilities from the binary gold-standard datasets supplemented with data from the output of a monolingual supervised English countability classifier (**EE$_{\text{BIN}}$(feat$_{\text{ALL}}$)**—see below). We then extract translation pairs from a bilingual dictionary (English–Dutch freedict version 1.1-1, containing 15,426 Dutch entries) and for each countability class, check for the existence of an English translation in the given countability class. If none of the English translations are classified as belonging to that countability class, we negatively classify the Dutch noun. In the event that no translation data exists for the Dutch noun or no countability data exists for the English translation(s), we classify the Dutch noun countability as unknown. Note that we map English plural only and bipartite nouns onto the Dutch uncountable class.

Transliteration-based classifier: EN$_{\text{BIN}}$(transliterate)

Transliteration-based classification also applies the observation that countability is frequently preserved under translation from English to Dutch, but does so in a resource-free manner. It takes a Dutch noun and simply determines if a countability-annotated word of the same spelling exists in English, and if so, transfers the countability directly across to Dutch. In all other respects, we implement the method identically to translation-based classification.

[6]We additionally built separate classifiers based on the outputs of the individual pre-processors, and also based on the multiclass data, but found their performance to be marginally inferior to that of NN$_{\text{BIN}}$(feat$_{\text{ALL}}$).

Cluster-to-cluster classifier: EN$_{\text{BIN}}$(cluster)

As observed above (§2.2), there is a strong correlation between the feature clusters used for Dutch and English. For example, co-occurrence with plural determiners is a strong indicator that the given noun is countable in both English and Dutch. At the same time, there is generally low correlation between individual unit features. For example, the English plural determiner *many* has no direct Dutch equivalent, and conversely, the Dutch plural determiner *sommige* has no direct English equivalent. The most straightforward way of aligning feature clusters, therefore, is through the (three) amalgamated totals for each one-dimensional feature cluster and some subset of the column and row totals for each two-dimensional feature cluster (e.g. for the PP feature, we align the totals for the singular and plural features but not the totals for each individual preposition independent of number). All values for the individual unit features are then ignored. In this way, it is possible to align 88 feature values, based on the output of the English and Dutch POS taggers.[7] Note that as part of the feature alignment, we take the negative log of all corpus frequency (*corpfreq*) values in an attempt to reduce the effects of differing corpus sizes in English and Dutch.

Feature-to-feature classifiers: EN$_*$(feat$_*$)

While we stated above that there is generally low correlation between individual unit features in English and Dutch, some unit features are highly correlated crosslingually. One example is the English singular determiner *a* which correlates highly with the Dutch *een*. Here, we can thus simply match the feature values onto one another directly. In other cases, a many-to-many mapping exists between proper subsets of a given feature cluster (e.g. the English determiner pair *each* and *every* correlates highly with the Dutch determiner pair *ieder* and *elk*), and alignment takes the form of feature value amalgamation in each language by averaging over the unit values and aligning the amalgamated values. A total of 466 unit feature values are amalgamated into 351 feature values, which are then combined with the 88 aligned total values from cluster-to-cluster classification

[7]All crosslingual feature-based methods were tested over the output of the POS taggers, the chunkers and the combined outputs of the three English and two Dutch preprocessors. Overall, there was very little separating the results, and the simple POS tagger generally produced the most consistent results.

for a total of 439 feature values. As for cluster-to-cluster classification, we evaluate feature-to-feature classification over the output of the English and Dutch POS taggers.

We implemented a total of 5 feature-to-feature classifiers: (1) **EN$_{\text{BIN}}$(feat$_{\text{ALL}}$)** makes use of all aligned features in the form of separate binary classifiers; (2) **EN$_{\text{MULTI}}$(feat$_{\text{ALL}}$)** similarly uses all aligned features, but in a multiclass classifier architecture; (3) **EN$_{\text{BIN}}$(feat$_{\text{DET}}$)** is based on only aligned determiner features, plus the aligned cluster totals; (4) **EN$_{\text{BIN}}$(feat$_{\text{PREP}}$)** is based on only aligned preposition features, plus the aligned cluster totals; and (5) **EN$_{\text{BIN}}$(feat$_{\text{PRON}}$)** is based on only aligned pronoun features, plus the aligned cluster totals.[8]

3.3 System combination

System combination takes the outputs of heterogeneous classifiers and makes a consolidated classification based upon them. It has been shown to be effective in tasks ranging from word sense disambiguation to tagging in consolidating the performance of component systems (Klein et al., 2002; van Halteren et al., 2001). In our case, we first take the outputs of all unsupervised (i.e. evidence-based) and crosslingual classifiers—a total of 12 classifiers—for each countability class (**EN$_{\text{BIN}}$(combined)**). We test the effects of system classification by way of 10-fold cross-validation over the 196 annotated Dutch nouns. This provides an estimate of the classification performance we could expect over unannotated Dutch noun data using the 196 annotated nouns as our sole source of annotated Dutch data. We also test combining the outputs of the 12 unsupervised and crosslingual classifiers with that of the **Alpino**-trained Dutch classifier (**E/NN$_{\text{BIN}}$(combined)**).

4 Results and Discussion

Classifier performance is rated according to classification accuracy (the proportion of instances classified correctly: **Acc**), precision (**P**), recall (**R**) and F-score$_{\beta=1}$ (**F**).

The baseline for each countability class is a majority-class binary classifier which simply classifies all instances according to the most commonly-attested class in the given dataset. Irrespective of the majority class, we calculate the

[8]Results for the multiclass classifier over feature subsets were found to be markedly worse than for binary classifiers.

Method	*Acc*	*P*	*R*	*F*
NN$_{\text{BIN}}$(majority)	.847	.847	1.000	.917
NN$_{\text{BIN}}$(evidence,all)	.551	.964	.488	.648
NN$_{\text{BIN}}$(evidence,chunk)	.510	.973	.434	.600
NN$_{\text{BIN}}$(evidence,POS)	.474	.970	.392	.558
EN$_{\text{BIN}}$(translate)	.948	.948	.331	.491
EN$_{\text{BIN}}$(transliterate)	**1.000**	**1.000**	.151	.262
EN$_{\text{BIN}}$(cluster)	.806	.957	.807	.876
EN$_{\text{MULTI}}$(cluster)	(.704)	.959	.837	.894
EN$_{\text{BIN}}$(feat$_{\text{ALL}}$)	.750	.983	.717	.829
EN$_{\text{MULTI}}$(feat$_{\text{ALL}}$)	(.704)	.959	.855	.904
EN$_{\text{BIN}}$(feat$_{\text{DET}}$)	.719	.966	.693	.807
EN$_{\text{BIN}}$(feat$_{\text{PREP}}$)	.755	.968	.735	.836
EN$_{\text{BIN}}$(feat$_{\text{PRON}}$)	.735	.952	.723	.822
EN$_{\text{BIN}}$(combined)	.873	.944	.904	.923
NN$_{\text{BIN}}$(feat$_{\text{ALL}}$)	.867	.961	.880	.918
E/NN$_{\text{BIN}}$(combined)	.903	.947	**.940**	**.943**
EE$_{\text{BIN}}$(feat$_{\text{ALL}}$)	—	.948	.972	.960

Table 1: Results for countable nouns

Method	*Acc*	*P*	*R*	*F*
NN$_{\text{BIN}}$(majority)	.638	.362	(1.000)	(.532)
NN$_{\text{BIN}}$(evidence,all)	.515	.423	**.930**	.581
NN$_{\text{BIN}}$(evidence,chunk)	.505	.414	.887	.565
NN$_{\text{BIN}}$(evidence,POS)	.628	.490	.718	.583
EN$_{\text{BIN}}$(translate)	.583	.583	.099	.169
EN$_{\text{BIN}}$(transliterate)	**.966**	**1.000**	.056	.107
EN$_{\text{BIN}}$(cluster)	.740	.692	.507	.585
EN$_{\text{MULTI}}$(cluster)	(.704)	.750	.592	.661
EN$_{\text{BIN}}$(feat$_{\text{ALL}}$)	.699	.750	.254	.379
EN$_{\text{MULTI}}$(feat$_{\text{ALL}}$)	(.704)	.822	.521	.638
EN$_{\text{BIN}}$(feat$_{\text{DET}}$)	.801	.758	.662	**.707**
EN$_{\text{BIN}}$(feat$_{\text{PREP}}$)	.776	.755	.563	.645
EN$_{\text{BIN}}$(feat$_{\text{PRON}}$)	.689	.708	.239	.358
EN$_{\text{BIN}}$(combined)	.791	.776	.593	.672
NN$_{\text{BIN}}$(feat$_{\text{ALL}}$)	.770	.783	.507	.615
E/NN(combined)	.812	.819	.609	.699
EE$_{\text{BIN}}$(feat$_{\text{ALL}}$)	—	.884	.907	.895

Table 2: Results for uncountable nouns

recall and F-score based on a positive-class classifier, i.e. a classifier which naively classifies each instance as belonging to the given class; in the case that the positive class is not the majority class (as occurs for uncountable nouns), the recall and F-score are given in parentheses.

We also provide an upper bound estimate of precision, recall and F-score based on a monolingual English countability classification task, with classifiers designed similarly to the monolingual Dutch classifiers (**EE$_{\text{BIN}}$(feat$_{\text{ALL}}$)**). In the case of English, the total number of feature values is 3,852, based on the concatenation of feature values from each of a POS tagger, chunker and dependency parser (Baldwin and Bond, 2003a). Our reason for choosing this as an upper bound is that it is based on moderate-volume, relatively noise-free training data and full feature correlation.

The classifier results are presented in Tables 1 and 2, broken down into the countable and uncountable classes. In each case, the best single value for each of evaluation metrics (other than the baseline and upper bound) is presented in **boldface**.

The first thing to notice is how much better the classifiers perform for countable than uncountable nouns. This is due to two factors: the relative occurrence of members of the two classes (as reflected in the majority class classification accuracies), and the relative volume of features correlated with each class. The relatively high baseline accuracy and F-score for countable nouns (.847 and .917) surpassed the performance of all classifiers other than the translation-based, transliteration-based, combined and monolingual classifiers. For uncountable nouns, on the other hand, appreciable gains over the baseline were observed for many of the systems. Results for countable nouns were relatively close to the upper bound results for the English monolingual classifier, whereas results for uncountable nouns were less competitive.

We have made the claim that, due to the lack of reliable training data in Dutch, crosslingual classification using English data is a viable option. This is borne out by the finding that the combined crosslingual classifier (EN_{BIN}(combined)) consistently outperforms the monolingual Dutch classifier in F-score, with the discrepancy being particularly noticeable for uncountable nouns. This finding is particularly striking given that the volume of Dutch training data is more than twice the volume of English data. Additional support comes from the analysis of the agreement between the system outputs the 196 hand-annotated nouns, recalling from §2.4 that the benchmark agreement for the **Alpino** data is 81.1%. The agreement for $\text{NN}_{\text{BIN}}(\text{feat}_{\text{ALL}})$ is 82.1%, that for EN_{BIN}(combined) is 83.2%, and that for E/NN_{BIN}(combined) is a respectable 85.7%. That is, all three methods produce countability judgements that are more parsimonious with actual corpus occurrence than the **Alpino** data, and the combined crosslingual classifier (EN_{BIN}(combined)) is superior to the monolingual classifier ($\text{NN}_{\text{BIN}}(\text{feat}_{\text{ALL}})$). Having said this, the combined crosslingual/monolingual classifier (E/NN_{BIN}(combined)) outperforms both the combined crosslingual classifier and the monolingual classifier, in which sense the **Alpino** data has some empirical utility. That is, we have shown that high-quality out-of-language English countability data is a stronger predictor of Dutch countability than medium-quality in-language Dutch countability data, but at the same time that the two are complementary.

There is very little separating the cluster-to-cluster and feature-to-feature classifiers. Given the high overhead in hand-aligning features in feature-to-feature classification, cluster-to-cluster classification would appear a low-cost, high-performance solution to the crosslingual countability task. Within the feature-to-feature classifiers, the results for the feature subsets are intriguing. We would expect that the determiner features should provide greater leverage than either the pronoun or preposition features, and this is indeed the case for uncountable nouns, where the determiner feature-based classifier returns the best F-score of all the classifiers. For countable nouns, however, the determiner features perform the worst of the three. Further research is required to determine the cause of this effect.

The results for the translation- and transliteration-based classifiers require qualification. Unlike the other classifiers, we do not get 100% coverage, as classification is possible only in the case that we have an English translation or transliteration with countability information. Strictly speaking, this diminished coverage should not be reflected in any of our evaluation metrics. In order to bring out this effect in Tables 1 and 2, we chose to base recall on the ratio of correctly-classified test exemplars to the number of positive-class exemplars, irrespective of whether the method is able to classify them. The F-score is thus proportionately low. If we were to base recall on the number of *classified* positive-class exemplars, the recall for the translation-based classifiers would become a perfect 1.000 ($\frac{55}{55}$) and 1.000 ($\frac{7}{7}$) for the countable and uncountable classes, respectively, and the corresponding numbers for the transliteration-based classifiers would be 1.000 ($\frac{25}{25}$) and 0.800 ($\frac{4}{5}$). That is, assuming we have English translation(s) for a Dutch noun or an English word of the same spelling, we get a very accurate estimate of the Dutch countability from the English countability data.

Finally, it is important to realise that these results are based on a limited test dataset (196 nouns) and that fuller evaluation is required to validate our findings. Also, our method relies crucially on the assumption that English and Dutch are closely-related languages, and its scalability to alternate language pairs remains to be determined.

5 Conclusion

We have presented several methods for classifying Dutch nouns as countable and/or uncountable on the basis of Dutch and English data. The classifiers depend on translation/transliteration data or linguistic features that were extracted from unannotated corpora. We compared a range of crosslingual English-to-Dutch classifiers with a monolingual Dutch-to-Dutch classifier, and found that the crosslingual classifiers outperformed the monolingual classifier to varying degrees. Based on this, we suggest that the optimal fast-track solution to Dutch countability classification is to use English data.

In future research, we are interested in the possibility of co-training via translation- and transliteration-based classification, as this seems to provide a means for automatically generating high-quality Dutch countability data to learn a monolingual classifier from.

Acknowledgements

The first author was supported by the National Science Foundation under Grant No. BCS-0094638 and also the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University. The second author was supported by the NWO Council for the Humanities, the Graduate School for Behavioral and Cognitive Neurosciences (BCN) and the Center for Language and Cognition Groningen (CLCG). The research was carried out within the framework of the PIONIER Project *Algorithms for Linguistic Processing*, which is funded by NWO (Dutch Organization for Scientific Research) and the University of Groningen.

We would like to thank Francis Bond, Ann Copestake, Dan Flickinger, Gertjan van Noord, Ivan Sag and the anonymous reviewers for their valuable input on this research, and John Carroll for allowing us to use RASP.

References

Keith Allan. 1980. Nouns and countability. *Language*, 56(3):541–67.

Harald R. Baayen, Richard Piepenbrock, and Hedderik van Rijn. 1993. *The CELEX Lexical Database (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Timothy Baldwin and Francis Bond. 2003a. Learning the countability of English nouns from corpus data. In *Proc. of the 41st Annual Meeting of the ACL*, pages 463–70, Sapporo, Japan.

Timothy Baldwin and Francis Bond. 2003b. A plethora of methods for learning English countability. In *Proc. of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 73–80, Sapporo, Japan.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proc. of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*, Toulouse, France.

Gosse Bouma, Gertjan van Noord, and Rob Malouf. 2000. Alpino: Wide coverage computational analysis of Dutch. In *Computational Linguistics in the Netherlands (CLIN 2000)*.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1499–1504, Las Palmas, Canary Islands.

Lou Burnard. 2000. *User Reference Guide for the British National Corpus*. Technical report, Oxford University Computing Services.

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL-2002)*, Taipei, Taiwan.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2002. TiMBL: Tilburg memory based learner, version 4.2, reference guide. ILK technical report 02-01.

Ralph Grishman, Catherine Macleod, and Adam Myers, 1998. *COMLEX Syntax Reference Manual*. Proteus Project, NYU. (`http://nlp.cs.nyu.edu/comlex/refman.ps`).

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics*, 27(2):199–230.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *Elements of Statistical Learning*. Springer-Verlag, New York, USA.

Dan Klein, Kristina Toutanova, H. Tolga Ilhan, Sepandar D. Kamvar, and Christopher D. Manning. 2002. Combining heterogeneous classifiers for word-sense disambiguation. In *Proc. of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, Pittsburgh, USA.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–23.

Tom M. Mitchell. 1999. The role of unlabeled data in supervised learning. In *Proc. of the Sixth International Colloquium on Cognitive Science*, San Sebastian, Spain.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proc. of the 2nd Annual Meeting of the North American Chapter of Association for Computational Linguistics (NAACL2001)*, pages 40–7, Pittsburgh, USA.

Leonoor van der Beek and Timothy Baldwin. 2003. Crosslingual countability classification: English meets Dutch. *LinGO Working Paper No. 2003-03*.

David Yarowsky, Grace Ngai, and R Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proc. of Human Language Technology (HLT) 2001*, pages 161–8, San Diego, USA.

Encoding and Presenting Interlinear Text Using XML Technologies

Baden Hughes, Steven Bird and Catherine Bow
Department of Computer Science and Software Engineering
University of Melbourne
Victoria 3010, Australia
{badenh, sb, cbow}@cs.mu.oz.au

Abstract

Interlinear text is a common presentational format for linguistic information, and its creation and management have been greatly facilitated by the development of specialised software. In earlier work we developed a four-level model and corresponding formal specification for interlinear text. Here we describe a suitable XML representation for the model and show how it can be rendered into a variety of convenient presentational formats. We conclude by discussing architectural extensions, an application programming interface for interlinear text, and prospects for embedding the interlinear model into existing applications.

1 Introduction

Interlinear text is a standard presentational form for displaying a source text aligned with variety of linguistic annotation phonological, morphological and syntactic analyses, glosses and translations. An example of Yidinj interlinear text is shown in Figure 1.

Interlinear texts can vary in the number of rows, the content and level of analysis for each row, and certain aspects of layout. However, as we survey a broad range of cases, we can observe consistent patterns in layout. These patterns have lead us to propose an abstract model of interlinear text which is sufficiently general that it encompasses the majority of cases based on a survey of print and electronic representations of interlinear text (Bow, Hughes and Bird 2003). As we will show here, the model has a natural representation in XML, which can be used to generate a variety of useful visualisations. We begin by describing the model in §2, then propose a suitable XML representation (§3). In §4 we discuss common presentation styles, and in §5 we show how they can be implemented using XSL. Finally, we report our work on a prototype (§6) and discuss future research (§7).

njundu	*wanjdjam*
you-SA	where-ABL
Where have you come from ?	

Figure 1: Yidinj Interlinear Text

2 The EMELD Interlinear Text Model

For the purposes of this discussion we adopt the following nomenclature. By *interlinear text* we mean a written record of an external linguistic event, consisting of a transcription aligned with linguistic analysis. A *line* of interlinear text refers to a row of transcription plus all the rows of analysis pertaining to it. Within the line, there is a *horizontal modality* of words and their analysis, plus there is the *vertical modality* of row elements to indicate the structure of the interlinear text. Additionally there is the *vertical alignment* of one row above the other; typically there is consistency from one block of rows to the next in the vertical alignment of these rows.

Within the context of the EMELD project (EMELD, 2000), Bow, Hughes and Bird (2003) proposed a four-level, hierarchical model of interlinear text consisting of Text, Phrase, Word and Morpheme levels. A *phrase* is a collection of transcription and its interlinear analysis arrayed across two or more rows, normally represented in interlinear text as beginning on a new line, and wrapping only if necessary. A *word* is a smaller collection of material (e.g. transcriptions, morphemes and glosses) that must be kept together on the same line. A *morpheme* is the smallest possible level of alignment between linguistic forms and their meanings. In contrast, a *text* is the highest level of structure corresponding to the external linguistic artefact being investigated. Just as it is possible for a word to contain a single morpheme, a phrase may contain a single word, and a text may contain a single phrase. Thus there is no prior commitment to the total length of this external linguistic artefact. According to this four-level scheme, the user has flexibility in the assignment of content to levels, and the decision may be influenced by both linguistic considerations (what is a 'word'? Is this one text or two?) and layout considerations (what should be kept on the same line in a display?). Additionally, at any particular level, analysis may be optionally omitted.

In this model, two rows are represented within a single node of the hierarchy if they are aligned with each other. For example, morphemes and their glosses, while being displayed on two different rows, are both represented at the Morpheme level of the hierarchy, given their 1:1 correspondence. The hierarchical model allows for the concatenation of morphemes into words, words into phrases, and phrases into text, as required by the interlinear text. It also allows for complex information structures to be represented in simple tree diagrams.

3 The XML Representation

We now turn to a discussion of the XML representation of the EMELD model. In this representation, the textual material has a number of levels, which can be considered nodes in an XML document. Each level consists of some content, and a sequence of children at subsequent levels. The content is given a user-defined type which documents its intended semantic interpretation.

```
<interlinear-text>
  <item type="user-defined">
    Content at the text level, such
    as metadata, or an unaligned
    transcription of the entire text,
    or a pointer to an unaligned audio file
  </item>
  <phrases>
    Nested XML content to represent the
    phrasal constituents of the text
 </phrases>
</interlinear-text>
```

This structure is repeated at each of the four levels, using the Yidinj example again:

```
<interlinear-text>
 <item type="title">A Yidinj Story</item>
  <phrase>
   <item type="number">98</item>
   <item type="gls">[When Damari eventually
did turn up at the fighting ground, Guyala
asked him:] 'Where have you [come] from?'
</item>
    <words>
     <word>
      <item type="txt">nundu</item>
       <morphemes>
        <morph>
         <item type="gls">you-SA</item>
        </morph>
       </morphemes>
     </word>
     <word>
      <item type="txt">wandam</item>
       <morphemes>
        <morph>
         <item type="gls">where-ABL</item>
        </morph>
       </morphemes>
     </word>
    </words>
   </phrase>
  </phrases>
</interlinear-text>
```

4 Interlinear Text Styles

Now that we have a model of the structure of interlinear text, we demonstrate its adequacy by showing how it can be used in generating a variety of layouts. (This logic is largely analogous to that used in conventional generative grammar: having analysed surface forms and proposed an underlying representation, it is incumbent upon us to provide the rules and constraints which can be applied to generate the original surface forms from the putative underlying representations.) Here we discuss a variety of presentation issues, before defining and demonstrating the mapping using XSL in the following sections.

4.1 Grouping of Content

Generalising from these examples, it should be possible to view an interlinear text with coarser-grained alignment, e.g. combining phrase-level items to form a single text-level item, or combining morpheme-level items to form a single word-level item. More generally, when a text is enriched by the addition of finer-grained segmentation, it should still be possible to view it with coarser alignment. In terms of the hierarchical model, this amounts to taking material from a lower-level set of nodes, concatenating it together, and storing it in a higher-level node. Just as we can ignore the low-level structures, we must also be able to ignore the high-level structures. For example, given an interlinear text it should be possible to extract its words or morphemes in order to construct a word-list. Therefore, it is a requirement on the rendering process that it can ignore aspects of the structure for the purpose of display and alignment.

4.2 Which Rows to Display

Interlinear texts have widely varying levels of detail, ranging from two rows to a dozen. When a text is enriched by the addition of another row of information, it should still be possible to view it in its original form without this extra row. Therefore, it is a requirement on the rendering process that it can omit specific rows from the display.

4.3 Row Styles

Font variation in form may reflect the preference of the author or the requirements of the publisher. It should be possible to view the rows of a given text in different styles without having to manually change the style of every item in the text. Therefore, it is a requirement of the rendering process that it can distinguish different types of rows and display them in user-specified fonts, typefaces, point sizes and so forth.

4.4 Ordering of Rows

There are some widespread conventions concerning the vertical arrangement of the rows of interlinear text. For example, morphemes are usually written underneath the corresponding words , and glosses are usually written underneath the corresponding morphemes. However, there are occasional exceptions to such patterns. At the text level we can observe considerable variation. In some cases, the notes on the text are placed after the phrases of the text, in other places the order is reversed. It should be possible to view a the rows of a given text in different orders without having to manually reorganise the layout of every item. Therefore, it is a requirement on the rendering process that it can distinguish different types of rows and display them in a user-defined sequence.

5 Rendering Interlinear Text with XSL

Our implementation of rendering is based on the Extensible Stylesheet Language (XSL) (Bradley, 2000), which can be used to transform XML documents into other formats. By choosing different stylesheets, or selecting different parameters for a given stylesheet, it should be possible to generate a variety of useful formats, whether for human consumption using a particular technology (e.g. conversion to HTML for delivery to a web browser), or for machine consumption (e.g. conversion to or serialisation of another XML format for delivery to another program).

The transformation performed by this model must accomplish two things: (i) convert the abstract XML representation into a format which specifies grouping, row ordering and styles, and (ii) convert the XML markup into the formatting instructions of some other language like PDF or HTML. The second of these can be further broken down into two stages: conversion to XSL Formatting Objects, and conversion to the delivery format. The full model is shown in Figure 2 below.

On the left we see the abstract representation, which is mapped using one of our stylesheets XSL_1 to a surface representation that fixes the grouping of items, ordering of rows, and so forth. A different choice of stylesheet will result in different groupings and orderings in the surface representation $\mathrm{XML}_{\mathrm{SR}}$. This format may be further transformed using third-party XSL stylesheets ($\mathrm{XSL}_{\mathrm{PUB}}$) to meet the requirements of publishers. We supply another transform $\mathrm{XSL}_{\mathrm{FO}}$ which converts the surface representation into a low-level representation using XSL formatting objects. Third party software can then be used to generate the format to be delivered to the end-user.

The key rendering challenge posed by interlinear text is line-wrapping. A line of interlinear text contains multiple rows, and these must be wrapped as a group, keeping words and their morphological analysis together on the same line. Thus, these multi-row constituents must be treated as indivisible entities. Formatting languages such as HTML (Raggett, Le Hors and Jacobs, 1999), DocBook (OASIS, 2003) and DSSSL (ISO/IEC, 1996) model a document as a collection of blocks (paragraphs, quotes, tables, lists) each containing lines of text. Critically, blocks must appear on a line of their own, or equivalently, lines cannot contain multiple blocks. In order to handle the line wrapping requirements of interlinear text, we need a language which permits inline blocks, such as TeX or XSL-FO (XSL Formatting Objects (Adler et al, 2003)). To facilitate flexible integration with web environment we have chosen to use XSL-FO.

Unfortunately, some XSL-FO engines do not handle inline blocks correctly at the present time (e.g. Apache FOP (The Apache Project, 2003) and XEP (RenderX, 2003)). After experimentation, we have found that XSL Formatter (AntennaHouse, 2003) correctly handles inline blocks, and so we have used it to generate the examples included below. For a discussion of XSL-FO engines and their various conformance levels, see Kimber (2002).

XSL processing of the abstract XML format to specify grouping, row-ordering and styles is quite straightforward. We give three examples to illustrate. The following template matches a phrase, and orders its constituent words before any content at the phrase level (such as a free translation).

```
<xsl:template match="phrase">
 <phrase>
  <xsl:apply-templates select="words"/>
  <xsl:apply-templates select="item"/>
 </phrase>
</xsl:template>
```

The following template matches an interlinear-text, ordering any content of type “title” before the constituent phrases, and ordering any other content (e.g. notes) afterwards.

```
<xsl:template match="interlinear-text">
 <interlinear-text>
  <xsl:for-each
   select="item[@type='title']">
  <title>
   <xsl:value-of select = "." />
  </title>
  </xsl:for-each>
   <phrases>
    <xsl:apply-templates
        select="phrases"/>
   </phrases>
    <xsl:for-each
      select="item[@type!='title']">
      <item>
        <xsl:value-of select = "." />
      </item>
    </xsl:for-each>
  </interlinear-text>
</xsl:template>
```

The following template matches a document and constructs a single interlinear text consisting of just the words (including any morphemes and glosses), and sorts them.

```
<xsl:template match="document">
 <document>
  <interlinear-text>
   <phrases>
    <xsl:for-each
     select="interlinear-text/phrases/
phrase/words/word">
    <xsl:sort select="."/>
     <phrase>
      <words>
       <xsl:copy-of select="."/>
      </words>
     </phrase>
    </xsl:for-each>
   </phrases>
  </interlinear-text>
 </document>
</xsl:template>
```

Using such templates we can generate a variety of document types for external processing (e.g. use by third-party software) or for delivery to end-users.

Further work on the XSL mapping is needed: to support common layout styles and to include additional rendering parameters; to permit display parameters to be stored within the abstract XML representation itself (or in a special XML format declaration file); and to model affixation in order that hyphens are introduced appropriately at morpheme boundaries.

6 Prototype

In demonstrating our model we have adopted a three level architecture which consists of an underlying data representation, a surface display format, and a variant display format. Essentially the processes involved in demonstrating the flexibility of this architecture are the conversion of the underlying data to a surface display, and then converting the surface display to a variant display.

6.1 Underlying Data

The underlying data is interlinear text structured according to our model and expressed in XML. This underlying data can be validated against an existing DTD or schema.

6.2 Surface Display

The surface display is a basic display format corresponding to traditional interlinear text which is enabled by the application of an XSL stylesheet to the underlying data. There are two types of surface display we have identified. The first is a simple type, namely direct application of a single XSL stylesheet to an underlying XML document. The second is more complex, including the parameterisation of user selected display input which in turn affects the XSL stylesheet and the corresponding display of the underlying XML document. These distinctions form the basis of the categorisation of functions.

6.3 Variant Display

The variant display is a customised display format which demonstrates the flexibility of manipulating the underlying data for different display purposes. For this demonstration we have identified a number of desirable variants based on common linguistic data structures.

6.3.1 Simple Display Types

We have identified two simple display types: (i) free translation as separate block and (ii) frame interface based expansion of free translation. In the first variant, we manipulate the surface display to format the free translation as a separate block of text from the interlinear content. In the second, we manipulate the surface display to provide the free translation in a separate frame from the interlinearization, and allow synchronised scrolling and linking between the segments of the free translation and the relevant interlinear segments.

6.3.2 Complex Display Types

We have identified a number of complex display types based on parameterised input. These are tree view or metastructural view; row reordering; optional row display; wordlist linkage; and concordance linkage. The *tree-view* or *metastructural display* essentially allows navigation of the interlinear text using a tree view display format. Individual branches of the tree can be expanded or compressed. This may be useful for structural analysis of the text. The *row reordering display* allows the selection of a preference for the order of the lines of interlinear text, eg display source text first, display source text last. This may be useful in evaluation contexts for back-glossing. The *optional row display* allows a selection of preference for how many interlinear lines are displayed. This may be useful for context where features of interest are identified in the interlinear text (eg syntactic vs morphological vs phonological annotation). The *word list linkage display* allows the selection of a particular word, and the corresponding display of interlinear content for that word. This may be useful for contexts where particular words are of intermittent interest and detected whilst browsing the source or translation text. The *concordance linkage display* allows the selection of a particular, and the corresponding display a list of of all other occurrences of the particular word within the text, including the surrounding words. Any context can be selected, and the complete interlinearization displayed for that context.

6.4 Implementation

We have implemented a prototype which supports the rendering of user-nominated display types. Our implementation is web based, with all user interaction occurring within the browser but reliant on embedded XML rendering capabilities and plugins to handle external application types such as PDF. Our implementation can be described as consisting of three modules, namely the user interface, the parameterisation logic, and the rendering engine.

The user interface is very simple, allowing the user to select the input text from a series of XML interlinear sources, then to select various display types, and finally to select an output format. Underlying each of these choices are a series of parameters, which are processed by a script to determine the display type and result type. These parameters then are passed to the rendering engine, which in combines the interlinear source and the option parameters to generate the appropriate output type which is then sent back to the browser for either direct display or display through a browser plugin.

At a technical level, the user interface itself is written in HTML, while the parameterisation logic is coded in PHP. We have experimented with two server side implementations - one is a wrapper based approach to a Java application executed in the shell; the other is using a Java servlet supported by Apache and Tomcat. (These alternative implementations reflect the lack of a fully-featured rendering engine which services both browser based rendering and plugin-based rendering.

There are a number of extensions to the prototype we hope to implement in the future, including a remotely instantiable rendering engine, the ability for users to contribute their own interlinear texts for rendering, the ability for users to control aspects of the display types, and a wider variety of output formats (eg. JPEG, SVG, or tree diagrams).

7 Future Research

Although we have developed a specification and prototype implementation for interlinear text, there are a number of areas which warrant further research. Here we identify and briefly discuss three of these, namely: architectural extensions; the development of an API for interlinear text manipulation; and the prospects for embedding of the interlinear text model with existing (and ultimately, new) tools. The creation of such tools which are designed for use by the linguist for creating, editing and publishing of interlinear texts represents a significant undertaking, but is based on the foundational work of analysing interlinear text structure, and expressing this in an open format.

7.1 Architectural Extensions

Having discussed the presentational flexibility that the XML-based specification provides, we can now turn to the corresponding architectural flexibility inherent in XML itself. We will consider linkages of interlinear text to higher level linguistic ontologies; as the subject for mining and retrieval in large corpora, such as the Web; and compatibility with other schemas for the representation of materials in an interlinear fashion.

7.1.1 Linguistic Ontologies

Interlinear text typically includes annotations regarding a wide variety of linguistic phenomena. Each annotator typically uses a different labelling inventory for such analysis, and thus automated cross-linguistic enquiry is made significantly more difficult through the disparate labelling of linguistic features. In order to leverage the large amount of annotated interlinear text which exists, there is a requirement for annotation to subscribe to a common ontology of linguistic concepts. Such a linguistic ontology (GOLD) has been defined in Farrar, Lewis and Langendoen (2002) and further refined in Farrar and Langendoen (2003). Our model provides ease of integration of such ontology-based annotation structures, and may provide a leverage point for theory-neutral cross-linguistic enquiry. Further work is required to fully exploit the power of a formally structured interlinear text with embedded ontological annotations.

7.1.2 Text Mining and Retrieval

Although we provide a model for encoding interlinear text and its subsequent manipulation, it is obvious that a vast amount of interlinear material already exists, some of which is in electronic form. In particular, interlinear text is published online through a variety of document types including lexicons, pedagogical materials, language recordings and transcriptions, language descriptions, grammars and scholarly papers about language. For the most part, this interlinear material is locked up in proprietary formats or is expressed only in a loosely structured fashion. Re-use of encoded interlinear text which exists in this form is desirable, but obstructed through formats which do not lend themselves to easy linkage. Some researchers, notably Lewis (2003), have embarked on efforts to identify interlinear text in these contexts, and to retrieve likely examples of interlinear text for re-use. In this particular context, the existence of a model into which existing interlinear forms can be easily translated is a desirable entity. Providing tools exist to manipulate interlinear text compliant with this new model may provide an incentive for linguistic data managers to explore conversion techniques for common proprietary or loosely structured formats. Assuming such conversion is carried out, the standardised body of interlinear text that results can in turn be queried directly for a variety of purposes.

7.1.3 Compatibility with Other Schemata

In our earlier work, we clearly delineated interlinear text from other materials which were expressed in an interlinear format. Models such as those proposed by Brugman (2003) use interlinear text as a representational medium for non-textual data, in particular for multimodal sources such as audio and video. The annotation of such data sources may require variation in the granularity of an interlinear model in order to capture the extra-linguistic dimension of the expression (eg gesture). Whilst we position our model as an interchange (and possibly archival format), we acknowledge that other annotation types require different structures in which to encode annotations. Already we have commenced investigations into the use of XML namespaces to allow both types of annotation and analysis to coexist within a single XML document. Software which is XML namespace-aware can then interpret rows of analysis according to the appropriate textual or multimodal models. Further work in this area is required to ensure tight integration and the development of appropriate conversion tools.

7.2 An API for Interlinear Text Manipulation

We have presented a data model but apart from our discussion of rendering we have not attempted to define the legal operations that construct, access and otherwise manipulate the structures. In future work we plan to define the data types and legal operations formally, independently of their possible XML and XSL representations. This will serve as the basis for object-oriented implementation, for the definition of application programming interfaces, and for research on suitable query languages. More generally, we need to consider interfaces to other data sources such as texts and lexicons.

7.3 Embedding Interlinear Functionality in Application Instances

The model presented here is a specialisation of the interlinear text model presented by Maeda and Bird (2000), and can be represented using annotation graphs. In future work we will implement a tool that is part of the Annotation Graph Toolkit (AGTK, http://agtk.sf.net), building on the existing InterTrans tool (Bird et al, 2002). This will require extending the XML format to support the specification of media file offsets which is likely to be trivial. In addition, translation tools which support conversion between native AGTK and our proposed model format for interchange will facilitate the import of existing data into this new tool.

8 Conclusion

Interlinear text is a highly pervasive data type in the linguistic domain. Although a number of tools have gained widespread acceptance for creating and editing interlinear text, the lack of an open and extensible model has resulted in annotated textual data in interlinear form being tied to particular implementations either as structural or presentation formats. In this paper we have sought to present open, extensible encoding and presentation mechanisms which allow the re-use of interlinear text in a variety of output formats. A significant advantage of adopting an XML-based structural encoding is that interlinear text can potentially be manipulated and queried systematically by any number of tools which subscribe to open standards, whether they have a linguistic lineage or otherwise.

References

Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Ridner and Steve Zilles. 2001. *Extensible Stylesheet Language Version 1.0* The World Wide Web Consortium. http://www.w3.org/TR/2001/REC-xsl-20011015/

Antenna House Incorporated. 2003. *XSLFormatter* Antenna House Incorporated. http://www.antennahouse.com/xslformatter.htm

The Apache Project. 2003. *Apache Formatting Objects Processor* The Apache Project. http://xml.apache.org/fop

Steven Bird, Kazuaki Maeda, Xiaoyi Ma, Haejoong Lee, Beth Randall, and Salim Zayat. 2002. *TableTrans, MultiTrans, InterTrans and TreeTrans: Diverse Tools Built on the Annotation Graph Toolkit*. Proceedings of the Third International Conference on Language Resources and Evaluation, Paris: European Language Resources Association, pp 364-370.

Catherine Bow, Baden Hughes and Steven Bird. 2003. *Towards a General Model of Interlinear Text*. Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

Catherine Bow, Baden Hughes and Steven Bird. 2003. *The EMELD Model for Interlinear Text* Manuscript in preparation.

Neil Bradley. 2000. *The XSL Companion*. Addison-Wesley.

Hennie Brugman. 2003. *Annotated Recordings and Texts in the DOBES Project* Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

DSSSL 1996. *Document Style Semantics and Specification Language* ISO/IEC 10179:1996

The EMELD Project 2000. *Electronic Metastructures for Endangered Language Documentation* http://www.emeld.org

Scott Farrar and Terry Langendoen. An Ontology for Linguistic Annotation. To appear in *GLOT International*.

Scott Farrar, William Lewis and Terry Langendoen 2002. *A Common Ontology for Linguistic Concepts*. Proceedings of the Knowledge Technologies Conference, March 10-13 2002, Seattle.

Baden Hughes, Steven Bird and Catherine Bow. 2003. *Interlinear Text XML and XSL Demonstration Handout*. Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

W. Eliot Kimber 2002. *Production Quality XSL-FO*. http://xml.coverpages.org/KimberProductionQuality-XSL-FO.html

William Lewis. 2003. *Migrating and Mining Interlinear Text*. Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA. http://www.emeld.org/workshop/2003/papers03.html

Kazuaki Maeda and Steven Bird. 2000. *A Formal Framework for Interlinear Text*. Proceedings of the Workshop on Web-based Documentation and Description, Philadelphia, USA; December 12-15, 2000.

Organisation for the Advancement of Structured Information Standards (OASIS). 2003. *DocBook* http://www.oasis-open.org/docbook

Dave Raggett, Arnaud Le Hors and Ian Jacobs 1999. *The Hypertext Markup Language Version 4.1* The World Wide Web Consortium. http://www..w3.org/TR/html4

RenderX. 2003. *XEP XSL Rendering Engine* RenderX Inc. http://xep.xattic.com/

9 Acknowledgements

This research in this paper has been supported by the National Science Foundation under Grant No. 0094934 (Electronic Metastructure for Endangered Languages Data).

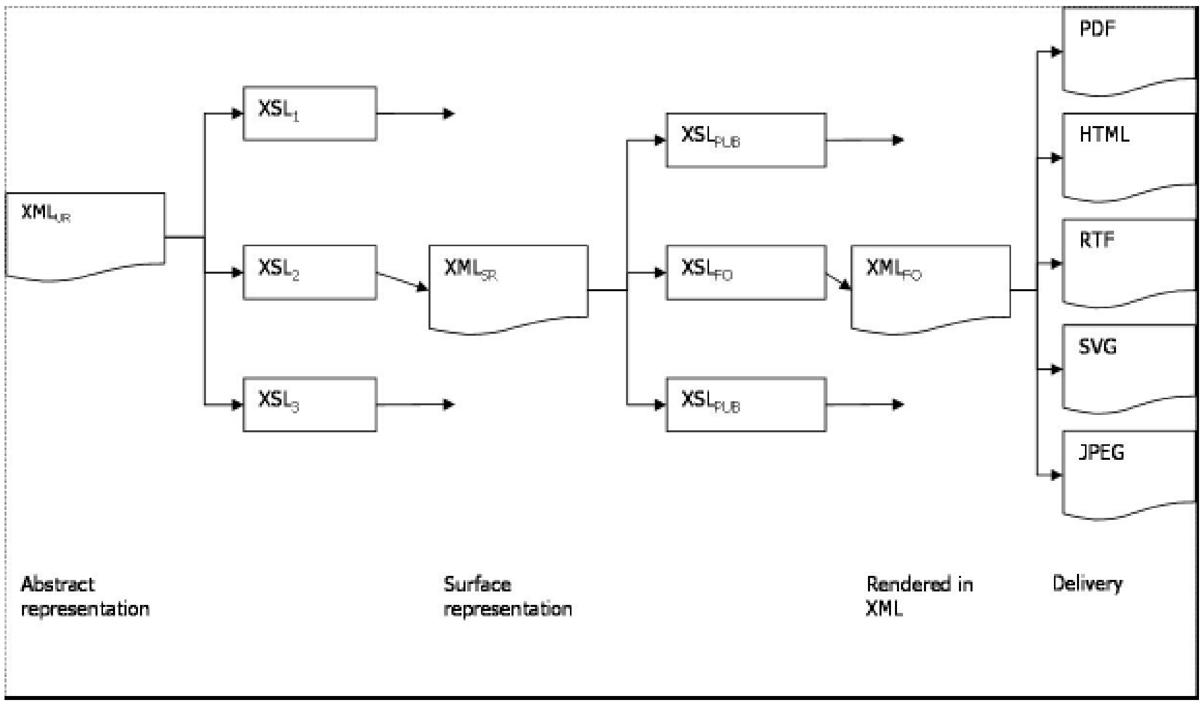


Figure 2: The Rendering Model

A queuing-theory model of word frequency distributions

Robert Munro
University of Sydney
rmunro@it.usyd.edu.au

Abstract

This paper describes a novel model for term frequency distributions that is derived from queuing-theory. It is compared with Poisson distributions, in terms of how well the models describe the observed distributions of terms, and it is demonstrated that a model for term frequency distributions based on queue utilisation generally gives a better fit. It is further demonstrated that the ratio of the fit/error between the Poisson and queue utilisation distributions may be used as a good indication of how interesting a word is in relation to the topic of the discourse. A number of possible reasons for this are discussed.

1 Introduction

When a given term occurs in a document, what is the probability that it will occur again, and how will this probability change when you have seen 1, 2 or n instances of this term? While NLP is a vast field, the majority of research is devoted to looking for intelligent ways to automate and speed up processes such as grammatical and/or semantic parsing, which a human could complete with an equivalent or much higher accuracy. Where NLP can exceed human accuracy is in extraction of patterns from large scale corpora (Banko and Brill, 2001). Lexical systems, such as term collocations, co-occurrences and frequency distributions, are one such area.

Typically, a word's occurrence across documents have been assumed to be Poisson in distribution. When this hasn't worked, a combination of two Poisson mixtures has been attempted with suggestions that when this fails combinations of three or more are appropriate (Bookstein and Swanson, 1974; Church and Gale, 1995).

1.1 Contribution of this paper

It is demonstrated here that the distribution of a queue utilisation (QU) is a more accurate representation for word frequency distributions than a Poisson distribution. A Poisson distribution captures the 'burstiness' of the frequency of a given word, otherwise known as the clustered-ness or self-collocation of a word. Here, it is shown that while the underlying assumption of burstiness holds, there are better representations for capturing this.

While it has previously been observed that words that best singularly describe the topic (variously labeled the 'interest words' or 'content words' of a discourse) are the words that *least* obey a Poisson distribution (Church, 2000), this has typically been used only as an observed trend, and without the search for what method best fits modelling this other distribution (if any).

The significance of the relationship between the fits of a Poisson and QU distribution is explored, and it is demonstrated that the level of disparity between the fits of Poisson and QU distributions can be used to extract words that best singularly describe participants in news topics with a very high level of accuracy.

This paper reports some preliminary results in using such a method on a large corpus of newswires, and discusses the results in relation to this register.

1.2 Outline

Section 2: background and related work.

Section 3: definitions of the equations use here.

Section 4: testing framework.

Section 5: results of testing.

Section 6: discussion & conclusions.

2 Background and Related Work

In terms of an attempt to describe the reasons for variance in the repetition of certain terms, the most recent work in this area is (Church, 2000) who defined the increased probability of a word occurring once it has already been seen 'positive adaptation'. It was the observation of the distributions described in this paper, along with the common trend of burstiness in network communications, that provoked the exploration of a queuing-theory model of word frequency distributions. There, and in (Church and Gale, 1995) the departure from a Poisson distribution were described as the result of 'hidden variables' relating to variation in register and author.

Word frequency distributions have been exploited in almost all tasks in computational linguistics and NLP, from subject boundary detection (Richmond et al., 1997), to information extraction/retrieval (Lynam et al., 2001; Pirkola et al., 2002; Jones et al., 2000).

The author's not aware of any previous word frequency distribution research using queuing theory models. Most previous research has used models based on information theory, presumably because it was to be used for information extraction/retrieval. Nonetheless, the corpus used here are newswires, so in the spirit of McLuhan, a model is investigated that is derived from the medium of communication.

3 Equations for probabilistic distributions

The goal of matching a word frequency distribution to a statistical equation is to find the optimal parameter value(s) for that equation, such that it fits the observed distribution with as little error as possible. There are different metrics and methodologies for calculating goodness of fit, including entropy measures, least squared regression and the practical effectiveness of the information when applied to some NLP task. Here, the chi-squared (χ^2) test is the metric used. This was appropriate for two reasons. It was the measure minimised by the optimisation algorithm (see below), and it is insensitive to error on the x axis, which is desirable as this represents an ordinal distribution and it is chiefly the divergence from the expected probability that is of interest.

3.1 Equations Used

These are the four equations that were tested here:

The Poisson distribution ($P(x)$):

$$\frac{e^{-\mu}\mu^x}{x!} \tag{1}$$

The Double Poisson distribution ($P_2(x)$):

$$\alpha\frac{e^{-\mu_1}\mu_1^x}{x!} + (1-\alpha)\frac{e^{-\mu_2}\mu_2^x}{x!} \tag{2}$$

The Queue Utilisation ($QU(x)$):

$$\rho^x(1-\rho) \tag{3}$$

The Double Queue Utilisation ($QU_2(x)$):

$$\alpha\rho_1^x(1-\rho_1) + (1-\alpha)\rho_2^x(1-\rho_2) \tag{4}$$

A weighted combination of a QU and Poisson distributions was also considered. It is desirable as only a Poisson distribution initially allows $P(x) < P(x+1)$ (for values of $\mu > 1$). Some brief testing confirmed it gave accurate fits, but it is not included in the testing and analysis here as:

1. It is difficult to find a theoretical justification for combining the two in this manner. If the only advantage is that a Poisson distribution allows $P(x) < P(x+1)$, then perhaps there exists some other equation that can model this in manner that is not so ad-hoc. Figures 3 and 6, which do possess this property, actually had less error in QU distributions, although it may take a close analysis of the graphs to see this. What is easier to see is that a combination of the two would definitely give a much better fit.

2. For an accurate comparison with the combinations of a single distribution type, it would be necessary to calculate the extent of overfitting that occurs, increasing and complicating the work described here.

3.2 Poisson

A Poisson distribution is often used to model systems where the *probability* of an observation is very low, but the *possibility* of an observation is very high.

In information retrieval, a double Poisson distribution is often favoured (Robertson and Walker, 1994). It is simply the weighted combination of two Poisson distributions.

Even with the additional parameters of the second μ and the weighting factor, a double Poisson distribution still fails to accurately capture word frequency distributions, but extending the equations to the combination of three or more Poisson mixtures will have the additional problem of the extra parameters to optimise. The most obvious disadvantage of this, other than the computational expense, is the potential for the resultant curve to overfit the data.

Church and Gale addresses this by suggesting Poisson mixtures (Church and Gale, 1995), which are an arbitrary number of Poisson distributions, allowing multiple values for μ, but varying according to a single density function, and therefore only adding one extra parameter. The assumption maintained here is that the underlying distribution *is* Poisson in nature, which is later noted as problematic (Church, 2000).

3.3 Utilisation Queue

In queuing-theory, a utilisation queue is an equation representing the probability of some queue having a given number of entities 'waiting' in it.

A queuing system is often described by the general M/G/1 model. Where:

ρ	=	The occupancy
μ	=	the average service rate
σ	=	the variance of the service time
c	=	$\sigma\mu$

The system is expressed by:

$$M/G/1(x) = \frac{\rho^x(1+c^2)}{x(1-\rho)} \quad (5)$$

Any queue may be described in terms of this equation, such as people waiting in line to buy tickets at a movie cinema, or the traffic on a computer network. The second of these, and its most common usage, is interesting in this context as it also represents a form of communication, albeit one a little more abstracted.

Within this, the model is often specialised to the M/M/1 model. It makes two assumptions about the data: both the queue arrival rate and the service rate are Poisson. Its relationship to a Poisson distribution is actually a little more involved, but the description of this is outside the scope of this paper and not directly relevant. This is given by:

$$M/M/1 = \frac{\rho^x}{x(1-\rho)} \quad (6)$$

The equation used here, the Queue Utilisation ($QU(x)$) model, is an equation describing the probability that an M/M/1 system has x items in it at a given point in time.

There are several immediate aspects of the QU equation that are appealing:

1. Simplicity. By Occam's Razor, if this were to model as accurately as a more complicated equation, then it would be the more desirable choice.

2. Extensibility. As there already exist, within queuing theory, extensions to the given equations to deal with various phenomena relating to density and limitations on queue size, these might be applied to the given situation.

3. Slower tail-off than Poisson. For large values of x, the tail-off is slower than for Poison, explicitly capturing the phenomena described both in, and by the title of, (Church, 2000).

A related equation called M/D/1 was tested but results indicated it was not appropriate for this task, but other variations of M/G/1 may have desirable properties.

Following a double Poisson distribution, a weighted combination of a two Queue Utilisation distributions was also considered.

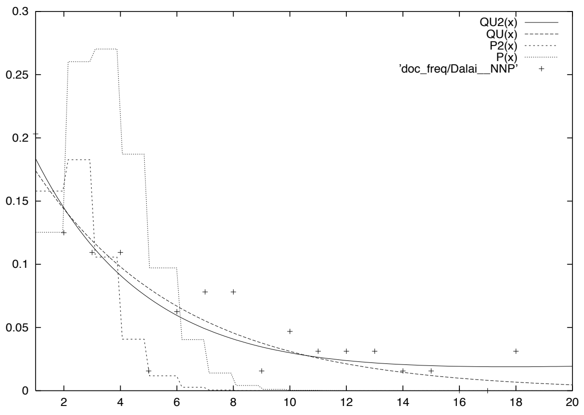


Figure 1: Fits for the term 'Dalai'

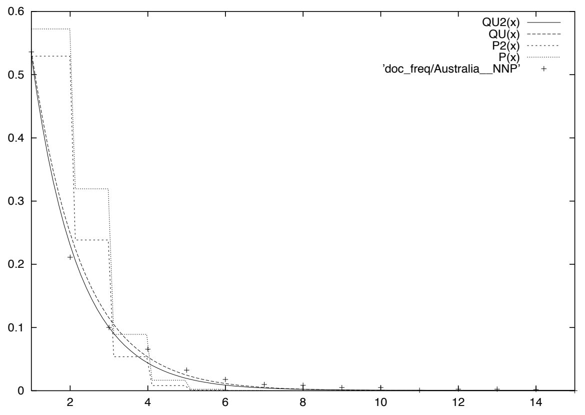


Figure 4: Fits for the term 'Australia'

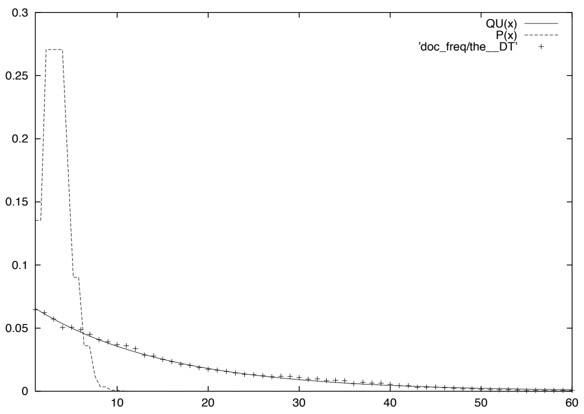


Figure 2: Fits for the term 'the'

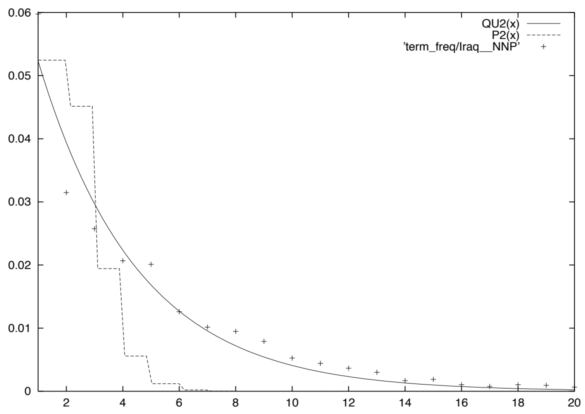


Figure 5: Fits for the term 'Iraq'

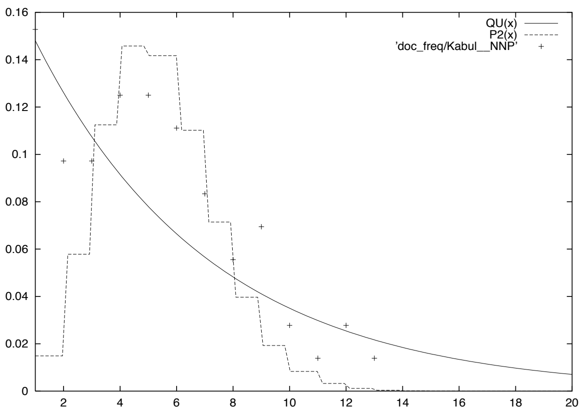


Figure 3: Fits for the term 'Kabul'

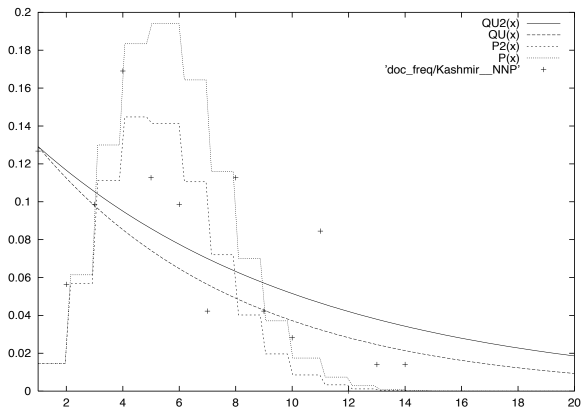


Figure 6: Fits for the term 'Kashmir'

4 Testing Framework

4.1 Corpus

The corpus used was a set of 50,000 *Reuters* newswires from the year 1996. This constituted approximately 15 million words. Newswires make an interesting register for this sort of analysis as each article typically addresses only one main topic or event, which is probably a better delimiter than using the raw number of words or sentences.

The words were marked up for part-of-speech by the brill-tagger (Brill, 1992). The terms considered were the word/POS pairs, so that a rough division between uses occurred. After some consideration, it was decided that the words should *not* be stemmed or lemmatised - either the distributions of all cases/tenses of a word are similar, or there are differences which may themselves be interesting to investigate.

To reduce noise, words that occurred with a frequency of less then 100 were omitted. Words that never occurred in a document with a frequency of 4 or greater were also omitted, as the differing curves of the equations used would not be emergent.

This resulted in 4935 word/POS tokens. The total number and frequencies of the occurrences of each of these were collated across all documents.

The recorded distributions were then normalised to be tested in two ways:

1. *document frequency*. All document frequencies were divided by the total number of documents containing the given frequency of the word. This is a 'true' normalisation, in the sense that the exact distributions are maintained.

2. *term frequency*. All document frequencies were divided by the total number instances of the term occurring with the given frequency. This will make the tokens with a relatively high level of self-collocation (positive adaptation) more emergent, relative to the document frequency, and is similar to the $tf.idf$ measure common to information extraction/retrieval and document classification.

POS	doc freq	term freq
noun	95.9%	99.9%
prop n.	93.5%	99.8%
verb	90.5%	100.0%
adj	91.8%	100.0%
adverb	90.9%	100.0%
all	94.3%	99.9%

Table 1: Percent of terms for which $QU(x)$ was a better fit than $P(x)$

POS	doc freq	term freq
noun	94.9%	94.6%
prop n.	91.6%	90.9%
verb	90.5%	89.9%
adj	89.6%	89.0%
adverb	89.8%	87.5%
all	93.1%	92.6%

Table 2: Percent of terms for which $QU_2(x)$ was a better fit than $P_2(x)$

4.2 Best Fits

The optimal models were found using gnuplot (Williams and Kelley, 1991), which implements the Marquardt-Levenberg algorithm for parameter optimisation (Press et al., 1992) . The equations were prevented from allowing $\mu < 0$ or $\rho < 0$, as this would give negative probabilities not appropriate here.

The success of the fits were gauged by the final sum of squares of residuals or the χ^2 result on the same term across the different equations.

Once set up, the entire process, from the single scan of the corpus, to the fitting of the equations, took only a couple of minutes to run on a current PC, the majority of this being the time taken for the 'double' distributions to converge.

5 Results

Results are reported for the relative success of the different distributions, given by the percentage of terms for which a given equation has a lower χ^2. These are given as totals and for the divisions into noun, proper noun, verb adjective and adverb parts-of-speech.

For unknown reasons, gnuplot was unable to converge the double Poisson for about 20 terms.

POS	doc freq	term freq
noun	68.7%	0.8%
prop n.	60.0%	1.5%
verb	60.1%	1.9%
adj	64.0%	0.8%
adverb	61.2%	1.1%
all	64.9%	1.1%

Table 3: Percent of terms for which $QU(x)$ was a better fit than $P_2(x)$

Comparisons were made between the fits of all Poisson / queue utilisation combinations. There were no significant results for which the single Poisson was more accurate than a double QU. The other results are given in Tables 1, 2 and 3.

The results demonstrate clearly that a queue utilisation model of word frequency distributions is a more appropriate representation for modelling term frequencies. The most impressive results are in Table 3, showing that even a single QU, with only one free variable, can provide a more accurate representation of the data than a double Poisson, which has two free variables plus the weighting factor.

A cursory glance at the accuracies in Table 3 would seem to indicate that modelling term frequency is less desirable than word frequency, despite the Table 1 accuracies being significantly better. However, looking at the best represented words by each, given by the largest ratio of a QU to a Poisson distribution shows the opposite. The top 1% of words for which $QU(x)$ was a better fit than $P_2(x)$ for term frequency is given in Table 4 (*my groupings*). These distributions are not emergent for the top 1% of words for document frequency, given in Table 5.

The trend for the term frequency distribution continued, with the next 1% of containing terms such as *Ansett, Chechnya, Iraq, Mother Theresa, Pope, Tibet* and *Tyson*, the one outlier being the determiner *the*. By contrast, the bottom 0.5% of terms, given in Table 6, are relatively mild and not particularly indicative of what the topics may be.

The ordering of terms for which a single $QU(x)$ was a better fit than a single Poisson corresponded well to the ordering of terms for the double Poisson, with the majority of terms in the top 1% also appearing in Table 4. Those that aren't in Table 4 are thematically related and include *Harridine, Holbrooke, Izetbegovic, rupiah, Srinagar* and *Zavgayev*.

Category	**Terms**
civil war/unrest:	Pol Pot, Ieng Sary, Rouge, Taleban, Kabul, Jalalabad, Afghan, Bossi, Kashmir, Albania, Liberia, Dalai, Hutu, Burundi
financial terms:	Grade, Prop, Select, asset-backed, min, m0, Jan-July, Level
(troubled) people:	Murtaza Bhutto, Portillo, Summers, Pen, Dutroux, Espy,
(troubled) currencies:	tolars, dinars, kroons, dirhams, bolivars, drachmas,
(troubled) companies:	Loewen, Michelin, Tractebel, Olivetti, Erste, Truck, DWT,
other:	inning, homer, vs, Manitoba, canola, Chernobyl

Table 4: Terms for which the $QU(x)$ gave the best fit, as compared to $P_2(x)$ for term frequency

Similarly, the other document frequency ratios gave orderings of ratios as relatively uninteresting as the set of words in Table 5.

Although the QU's best described terms were proper nouns and nouns, and the worst more of a mix of adverbs and verbs, the various parts-of-speech were represented about equally as well as each other. A minor exception can be seen in Tables 2 and 3 where the double Poisson was a little less able to improve the representation of some nouns.

In brief, while a document frequency model more accurately described the distribution of terms, a term frequency model would seem to be much more useful for comparing with Poisson to extract topics from a very large volumes of information.

6 Discussion and Conclusion

There are some proper nouns that serve as more static exophoric referents than others. The ones in Table 4 are among the most strict, with the notable absence of even the most common colloca-

from, which, with, it, an, then, spokesman, was, to, in, and, of, a, by, the, seek, not, but, for, bullish, discounts, before, we, trader, added, at, day, will, as, casualties, been, up, because, durum, cruise, futures, its, on, between, sharply, firmer, be, after, hand, this, have, wheat, while, nation, now

Table 5: Terms for which the $QU(x)$ gave the best fit, as compared to $P_2(x)$ for document frequency

Nigerian, believed, strength, steady, beyond, treasury, none, midday, critics, relatively, approved, thought, began, begin, making, delayed, expect, getting, accept, finished, possibly, failed, mainly, once, recently

Table 6: Terms for which the $QU(x)$ gave the worst fit, as compared to $P_2(x)$ for term frequency

tions of first name or title. This means that the terms in Table 4 are also among the least polysemous, and therefore the most pure in their distributions in this regard. The same can be said for the demonstrative *the*, also an explicit referent although typically as an anaphoric reference in written text.

The 'financial' words in Table 4 can be explained simply in terms of their being generic economic terms that are commonly tabulated or used in lists. They are not discussed further.

It is telling that in the top 1% of best represented terms, all the named entities are of persons, places or organisations involved in some sort of conflict. As a topic extraction task this is quite an achievement, but to a certain extent it can also be seen as a product of the register.

Assuming a word is a good indication of topic, there are two broad reasons that the word may also exhibit high 'adaptation':

1. The nature of the subject matter is complex, requiring a longer article and the introduction of more themes/participants.

2. The word represents one of multiple participants, and the explicit repetition of the word is chosen instead of the use of anaphora, in order to maintain cohesion.

A third reason, that is worth speculating about but is hard to confirm in a study necessarily abstracted from the text itself by scale, is that the phenomenon is capturing an underlying subjectivity. In news reporting, the use of narratives of conflict are not limited to articles describing military conflicts, often being used in sports reporting. Here, only the terms *inning* and *homer* evidence this. This may simply be because sports reporting is known to favour the continuity of Subject, and therefore *is* able to make more use of anaphora, while maintaining Subject in the reporting of a military conflict is more likely to be viewed as Subjectivity and is therefore avoided, at least grammatically. But perhaps there is something about a term like *Taleban militia*, which, independent of context, makes a news reporter desire to repeat it more than a term like *St Louis Cardinals*, and that this is, in part, a measure of attitude.

Relating to the first reason above, in the reporting of conflict, the explicit attributing of actions and possessions to entities is common, and so repetition will not just be as group/phrase heads but as possessives, *Burundi's army*, and Classifiers, *Burundi radio*.

While these properties will not always hold, it is the emergence of these patterns over large scales that gives these words greater positive adaptation. Even if the results indicated by Table 4 are mostly a product of the lack of anaphora and ellipses, rather than their 'content' or newsworthiness, they are still quite an accurate set of terms for describing topic for any technique that *doesn't* look at grammatical systems.

As stated, a Poisson distribution is often used to model systems where the probability of an observation is very low, but the possibility of an observation is very high. In reality, if a word relates closely to the topic, then the probability of observation is high, but the possibility of (coherent) uses becomes low (in a report on Afghanistan, the term *Kabul* cannot not easily be made a referent to anything other than a specific city). Assuming that for a word to be commonly repeated more than three or four times in a document indicates that it is likely to relate closely to the topic, then for all such words it is to be expected that a Poisson distribution is inadequate.

A brief manual inspection of the terms that occur with a total frequency greater than 100, but never with a document frequency of four or greater (there were about 1700 additional words meeting this criteria) showed that, beneath the noise, these were disproportionately likely to be Epithets, realised by either an adjective or a gerund. This is not surprising, as when a term is functioning as the property or evaluation of an entity, not part of the entity itself, there is rarely the need to remind the reader/listener of this property/evaluation in a short discourse. An additional experiment with words occurring with a maximum document frequency of three was also conducted, giving a further small advantage in overall fit to the QU distribution, but no significant additions to the word lists already given.

The term *Noriega* (Church, 2000) appeared with frequency less than 100, and was therefore omitted from the experiments. This shows the other known 'burstiness' property of term frequencies, that they are also temporal. The articles used here were deliberately taken from a small window of time to negate this, but it would be interesting to see if the same model could be used for modeling temporal distributions independently of, or in addition to, the frequency distributions.

As conflict discourses feature multiple participants, the investigation of co-occurrence would be interesting, as too would be the investigation of collocations/n-grams.

For the most part, the part-of-speech tags alone would have been enough to cluster the various words into the groupings given in Table 4, but with information about the words, such as the error and discovered optimal ρ, it would be interesting to see what additional tendencies could be discovered and/or exploited in a variety of NLP tasks.

Acknowledgements

Thankyou to the ongoing support of my supervisors, Geoff Williams and Sanjay Chawla.

References

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Meeting of the Association for Computational Linguistics*.

A. Bookstein and D. Swanson. 1974. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25(5):312–318.

E. Brill. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*.

K. Church and W. Gale. 1995. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190.

K. W. Church. 2000. Empirical estimates of adaptation: The chance of two Noriegas is closer to $p/2$ than p^2. In *Proceedings of The 18th International Conference on Computational Linguistics (COLING-2001)*.

K. S. Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments - part 2. *Information Processing and Management*, 36(6):809–840.

T. R. Lynam, C. L. A. Clarke, and G. V. Cormack. 2001. Information extraction with term frequencies. In *Proceedings of the Human Language Technology Conference (HTL'01)*.

A. Pirkola, E. Leppänen, and K. Järvelin. 2002. The RATF formula (Kwok's formula): exploiting average term frequency in cross-language retrieval. *Information Research*, 7(2).

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C*. Cambridge University Press, 2 edition.

K. Richmond, A. Smith, and E. Amitay. 1997. Detecting subject boundaries within text: A language independent statistical approach. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*.

T. Williams and C. Kelley. 1991. Gnuplot - an interactive plotting program.

T-Code Compression for Arabic Computational Morphology

Jim Yaghi and **Mark R. Titchener**
Department of Computer Science
University of Auckland.
jyag001@ec.auckland.ac.nz, mark@tcode.auckland.ac.nz

Sane Yagi
Department of Linguistics
University of Sharjah.
saneyagi@yahoo.com

Abstract

It is impossible to perform root-based searching, concordancing, and grammar checking in Arabic without a method to match words with roots and vice versa. A comprehensive word list is essential for incremental searching, predictive SMS messaging, and spell checking, but due to the derivational and inflectional nature of Arabic, a comprehensive word list is taxing on storage space and access speed. This paper describes a method for compactly storing and efficiently accessing an extensive dictionary of Arabic words by their morphological properties and roots. Compression of the dictionary is based on T-Code encoding, which follows the Huffman encoding model. The special characteristics inherent in the recursive augmentation method with which codes are created allow compact storage on disk and in memory. They also facilitate the efficient use of bandwidth, for Arabic text transmission, over intranets and the Internet.

1 Introduction

1.1 Challenges of Arabic

Arabic poses a formidable challenge for computational linguists due to its derivational nature. Word generation requires moulding three- and occasionally four-consonant roots into a range of morphosemantic template patterns, where the root radicals intersperse between a templates' letters to produce a new word with a new meaning that still shares the basic meaning of the root. Often these templates augment the root by lengthening its medial radical, inserting a long vowel between the radicals, and/or adding consonantal prefixes. The generated words are what is termed '*stems*' in the English language, but they are not actual words; they are mere semantic abstractions. To become actual words, the stems are moulded into morphosyntactic patterns that will indicate whether a word is a verb or noun, present or past, active or passive voice, etc.

The process of root extraction from actual words, on the other hand, is not a simple reversal of the process of word generation because the root radicals would have been disguised by the application of morphological patterns.

Although Arabic morphology is systematic, it has remained a challenge to produce, for example, useful spell checkers, grammar checkers, search engines, and indexers that are not based on exact matching. The missing ingredient at the base of this problem is an accurate root-based morphological analyser. Spell-checkers, normally, do not contain a large enough word list to accommodate the inflectional variation words undergo when affixed because these may run in millions. Few grammar checkers exist for Arabic, because it is difficult to parse a sentence if its words are not correctly interpreted by a morphological parser. Various types of Arabic search engines are significantly impaired because of the inability to find character-to-character correspondence between search terms and variant match items such as differing tense, voice, person, number, or gender.

1.2 T-Code Technique

Finite State Transducers (FSTs) have been established as a standard way to encode morphological alterations (Karttunen et al., 1997). However, FSTs are normally compiled from rules written in a special FST generator language. FST compilers like PC-KIMMO (Antworth, 1990) and 2lc (Karttunen, 1993; Karttunen and Beesley, 1992) use a specialised language to generate lexical transducers. On the other hand, our implementation uses the standard PERL regular expressions (Friedl, 2002) but in a specialised manner.

Beesley (2001) describes a system that generates FSTs using *2lc* for lexical transformations of Arabic words. When generating words, the system uses the compiled FSTs to achieve morphological and phonological letter alterations and then uses them in reverse to perform derivation. Our approach uses, like Beesley, the compiled FSTs for word generation, but it does not use it for root derivation.

Our approach produces a list of Arabic stems, inflected affixed forms along with their roots and complete morphological classifications; this list facilitates the direct regeneration of words. Our root derivation technique requires this extensive list or dictionary of stems to be stored in a search-friendly manner.

The dictionary of stems would ordinarily occupy a large amount of disk storage space, but we propose here a technique that finds an acceptable balance between compression and lookup speed, T-Code (Titchener, 1984). T-Code compression is similar in style to Huffman encoding, as T-Codes are a subset of all possible Huffman code sets (Gunther, 1998). T-Code has the advantage of statistical synchronisation, or the ability to self-synchronise (Titchener, 1997), making it ideal for transmission over networks, especially where information loss is inevitable (eg., wireless networks).

We begin with basic roots and morphosemantic, morphosyntactic, and inflectional affixation rules to generate all possible stems. After some simple affix removal rules are applied, any valid lookup word should be found in the comprehensive list of stems. Internally, Arabic words are encoded with their roots and morphological classification so that the original word may be regenerated when needed.

In this paper, we discuss the system we have built for verb generation which can be used either in whole or in part for root-to-word and word-to-root lookups. We begin with a description of how the word list was generated, followed by a discussion of the dictionary format and how it was compressed, we then describe how the compressed dictionary was searched and decoded, and finally conclude with some suggestions for simple applications.

2 Word Generation

2.1 Word Data Creator Environment

The Word Data Creator Environment (WCE) was built to assist in creating and debugging the generation database. This software provides a graphical user interface facilitating data entry and experimentation.

WCE allowed us to edit the *MainDictionary Table*. For each entry, we were able to supply a root radical, a root classification identification number, and two numbers identifying the morphosemantic pattern and a morphosyntactic variant that a derived word would follow. Unlike traditional root-type classifications in Arabic morphology, our root classifications identify a root by the type and location of alterable letters it contains, for example, و (w), ي (y), and ء (')[1]. Alterable letters are those that usually undergo rule-based transformation if followed or preceded by certain other letters.

In addition to entry editing, WCE allowed us to edit related template entries from the *Templates Table*. A *Templates Table* entry is indexed by a pattern and variant identifier and a tense and voice combination. Every entry specifies a general template string which, for the given voice and tense, causes derived words to have a certain meaning. Entries also identify a set of inflection and spelling transformation rules and an affix list number. Transformation rules are dependent

[1]For readability, this paper uses Buckwalter's Arabic orthographical transliteration scheme (http://www.cis.upenn.edu/~cis639/arabic/info/translit-chart.html).

on a combination of the template string letters and the root radicals. The template strings of each entry are, in fact, the combined result of a morphosemantic and a morphosyntactic pattern, transformed for the tense and voice of the entry. The possible tenses are past and present, the voices are passive and active, and the modes are indicative and imperative.

Affix lists, which were also editable from within WCE, contain patterns for generating 17 different morphosyntactic forms specifying combinations of gender, number, and person for each voice and tense. Both affixation and transformation rules are specified using the language of PERL regular expressions.

2.2 Word Generation Engine

Within WCE is an implementation of the Word Generation Engine (WGE), which allowed us to debug our classifications and transformation rules, and to ensure the correct spelling of generated words. While making modifications to root radicals, word classifications, template strings, and transformation and affixation rules, we were able to preview the result of any of the 17 word types on the main screen for the selected *MainDictionary Table* entry.

The three components, *Stem Transformer*, *Affixer*, and *Slotter*, activated in sequence, make up WGE. *Stem Transformer* applies the appropriate transformation rules to the stem template, *Affixer* adds an affix to the transformed template, and finally *Slotter* applies the transformed radicals to the affixed template to produce the final affixed word.

WGE begins with a stem ID from the *MainDictionary Table* as input. The root and entry associated with the stem ID are used to identify the radicals of the root, the stem template string, the set of transformation rules that apply, and an affix list.

Stem Transformer is applied incrementally using radicals, a template string, and one transformation rule per pass, as in Figure 1. The output of each pass is fed back into *Stem Transformer* as a modified template string and modified radicals, along with the next transformation rule. When all rules associated with the template are exhausted, the resultant template string and radicals are output to the next phase.

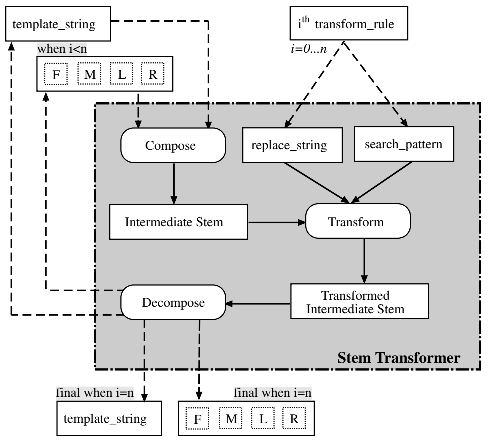


Figure 1: Stem Transformation Phase

A template string marks the positions at which radicals belong in the template by using the Roman letters F, M, L, and R. These may be viewed as the variables in the template; all other characters are Arabic constants. Stem Transformer begins by inserting the root radicals directly after their position markers. For example, a template, إ Fْ تَ Mَ Lَ (<i**F**ota**M**a**L**a)[2], with a radical set {ر ،ك ،ذ } ({*,k,r}), becomes إ Fذْ تَ Mكَ Lرَ (<i**F***ota**M**ka**L**ra). The result is then decomposed back into the template form, and the root radicals are updated if altered. For the same example, the stem template is transformed by an ordered sequence of rules {1,12}. The text of rule 1 is: F(.)([َ ُ ِ ّ ْ]*)(ت) F$1$2$1. The first part specifies the match pattern and the second specifies the replace string. Rule 1 removes the infix letter ت (t) and replaces it with a copy of the first radical which should directly follow the radical's diacritic. The result is the string, إ Fذْ ذَ Mكَ Lرَ (<i**F***o*a**M**ka**L**ra).

Stem Transformer concludes by decomposing the updated template into template text and radicals. The altered template and radical set are then passed back into *Stem Transformer*, where another rule from the rule sequence may be applied. For the example above, the decomposed template becomes إ Fْ ذَ Mَ Lَ (<iFo*aMaLa), while the root radical set remains unchanged. During this second pass, *Stem Transformer* uses the altered template and radical set

[2]The letters F, M, L, and R in bold are radical position markers, not transliterations.

as input together with rule 12, whose text is, ([FMLR]?)([^ َ ُ ِ ّ ْ]*)([َ ُ ِ ّ ْ]*)([FMLR]?)(\2) $1$2ّ. Rule 12 is a gemination rule, and uses a backreference in the search pattern, in order to match any repeated letter. With its replace string, the second of the duplicate letters is replaced by the gemination diacritic, /ّ/ (~). The decomposed result is the template, إFَّMَLَ, and the untransformed radical set, {ذ، ك، ر} ({*,k,r}), which can produce the word إِذَّكَرَ (<i*~akara).

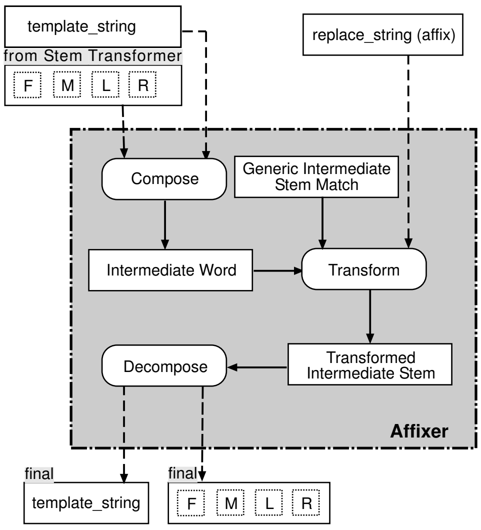


Figure 2: The Affixer Phase

In brief, the final output of *Stem Transformer* is a root-transformed template and a template-transformed radical set. These outputs are used as input to the affixation phase which succeeds stem transformation. *Affixer*, which is applied iteratively on the result of *Stem Transformer*, outputs 17 different affixed morphosyntactic forms for every stem. *Affixer* is run with different replace strings specific to the type of affix being produced. It modifies copies of the transformed stem from the previous phase, as in Figure 2. For example, إFَّMَLَ is passed to Affixer, with radical set, {ذ، ك، ر} ({*,k,r}), and the past active feminine singular affix replace string, $1$6M$7َL$11ْت. Figure 3 shows the generic transformed-template match string and indicates the back-reference groupings, which are used in the replace string for the affix. The result of applying the affix transformation above is the affixed template string, إFَّMَLَتْ (<i**F**~a**M**a**L**ato).

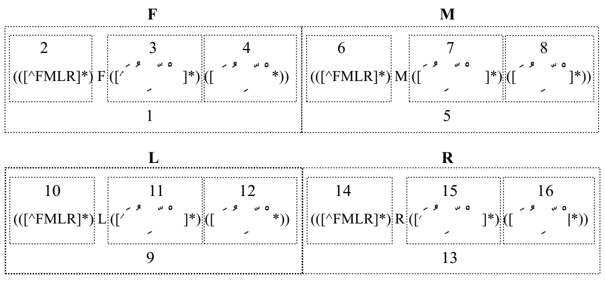


Figure 3: The generic transformed-template match string

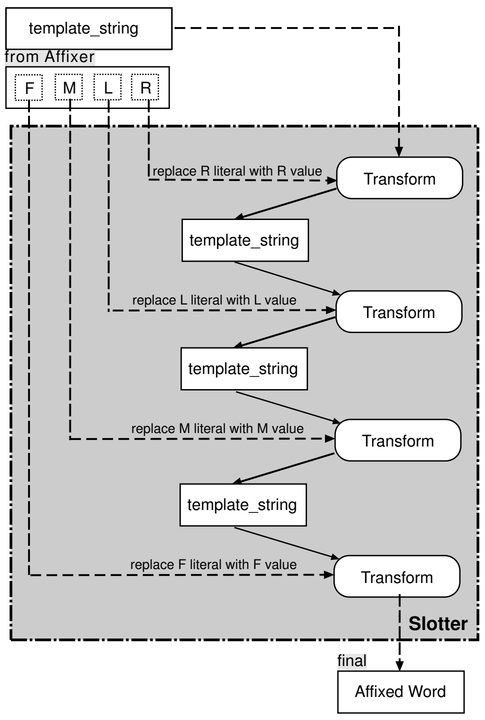


Figure 4: The Slotter Phase

In *Slotter*, the last stage of word generation, transformed radicals replace the Roman position markers in the transformed template, to produce the final form of the word. For the example above, the result is إِذَّكَرَتْ (<i*~akarato) which is the past active feminine singular form of the word.

3 T-Code Encoding

Using a format that allows searching the database, we output an alphabetically sorted list of each of the 2 million words that WGE generated. Since diacritics are optional in written Arabic, we wanted to facilitate the matching process by having the possibility of ignoring diacritics or only matching those diacritics that a search item specifies. In order to achieve this, we indexed our list for lookup using bare words, words without diacritics. For each entry, we included the root, template, and affix type identifiers as numbers. This gave the capability of generating the actual word after lookup in order to pinpoint an exact diacritic match if necessary.

Indexing the complete word list from WGE and storing it in a disk-based B-Tree data structure yields files larger than 100 MB[3] Since our dictionary only represents the verbs of Arabic, adding the nouns would at least double its size; therefore, it would be advantageous to keep the dictionary's disk size minimal.

T-Code encoding, like Huffman encoding, is a variable length coding scheme. The basis of T-Code text compression is that shorter codes are assigned to frequently repeated items. Since uncompressed text is normally represented by fixed length codes in software, T-Code is capable of achieving a large compression factor for text because it has low entropy. For the word database produced by WGE, a great amount of redundancy exists since the 2 million words are based on only 5,500 verb roots. T-Code has the advantage of self-synchronisation; that is, a series of bits from a code will only be recognised as being members of the T-Code set if they constitute a valid code word. If a series of bits does not belong to the T-Code set, it will not be valid until all the bits of the code arrive. This is useful because no additional code length information needs to be stored in the data.

The T-Codes used to encode the database are obtained by first calculating a target distribution of code bit-lengths, then creating an adjusted T-Code distribution based on the target, and finally assigning the shortest codes to the most frequent data items.

3.1 Calculating a Target Distribution

A target distribution for the dictionary database was calculated using the frequencies of its unique items. The equation below was used to calculate the code's target bit-length ℓ for each data item i in the database using the item's average frequency $\bar{f}_i$.

$$\ell_i = -\lceil \log_2 \bar{f}_i \rceil, i = 0...n$$

We grouped the frequencies of unique root, template, and affix type identifier numbers for each word entry. Additionally, a slightly different frequency count for the letters of the lookup words was performed in order to take into account their compressed form. Special attention was given to the compression of the low entropy lookup words whose efficient access is essential.

Original		Letters Counted	
Entry	**Transliteration**	**Entry**	**Transliteration**
اب	Ab	اب	Ab
ابا	AbA	..ا	..A
ابابا	AbAbA	...با	...bA
اباتت	AbAtt	...تت	...tt
اباتمتم	AbAtmtm	....متم	....mtm
اباتنتن	AbAtntn	....نتن	....ntn
اباث	AbAv	...ث	...v
اباثن	AbAvn	....ن	....n
اباج	AbAj	...ج	...j
اباجا	AbAjA	....ا	....A
اباجن	AbAjn	....ن	....n
باجوا	bAjwA	باجوا	bAjwA

Table 1: Eliminating redundancy by not counting repeated letters.

The bare words forming the lookup entries have a one-to-many relationship with actual words. That is, many different generated words with diacritics may become the same lookup entry when diacritics are removed. Therefore, if it is possible to distinguish between one bare word and the next, repetition of lookup entries is unnecessary. A bit-skip field is used in the encoded database to mark the end of an entry; details of this and the encoded database format are discussed in Section 3.2. During this phase, we were only concerned with the frequency of the letters of the lookup items in the final database, so unique entries had their letters counted only once.

Another source of redundancy in lookup items appears in their alphabetically sorted form. Often, an entry shares initial letters with following

[3]The file size being so large is explainable by the fact that Unicode UTF-16 uses 16-bits per Arabic character (Consortium, 2003), which causes output to be twice as large as it may have been for Roman characters.

entries. While the dictionary format handles this, calculation of a target distribution only counts letters not sequentially shared between consecutive entries, as may be seen in Table 1.

Code Length	Target Frequency	Modified Target	T-Code Distribution
5	9	5	5
6	5	9	9
7	-	-	0
8	2	2	2
9	-	-	0
10	4	4	4
11	3	3	3
12	9	9	9
13	10	10	10
14	2	2	2
15	-	-	0
16	16378	3836	3836
17	2750	7232	7233
18	-	8059	14159
19	-	-	27308
20	3	3	52009

Table 2: A T-Code distribution from the target distribution for the dictionary.

3.2 Encoding the Dictionary

A T-Code distribution was calculated based on the target distribution, as in Table 2. Its codes were created and sorted from shortest to longest then assigned to the unique data items of the database in order of most frequent to least frequent. The uncompressed dictionary's data items were then T-Code encoded.

Figure 5 depicts the encoded dictionary structure. A header is used to identify the positions of the start and end of the encoded data. The T-Code encoded data is represented as a continuous bit stream written in byte-sized units.

3.2.1 Indexing and Accessing the Dictionary

Access to the dictionary is required to be sequential. Without a proper indexing system lookups would be inefficient, having potential complexity of order *O(N)*. To facilitate efficient lookups, a simple first letter lookup was used to give direct access to the byte position of the first entry using the first letter of the lookup word.

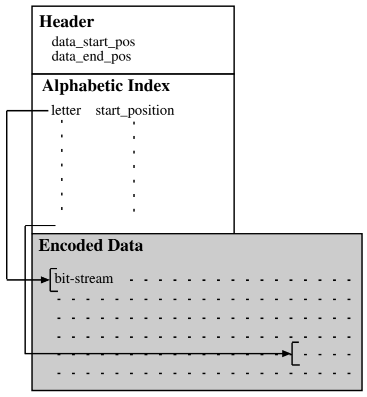


Figure 5: The encoded dictionary structure.

While the first-letter-lookup gives a reasonable efficiency advantage, the rest of the lookup process is required to sequentially read the entries starting with the first letter. In order to address this, we added a two-byte fixed width field at the start of every entry, and distributed their bits as in Figure 6. An example in Table 3 illustrates how the fixed width fields are used.

Pos.	Entry	Transliteration	Next Pos.	Shared
0	اب	Ab	11	0
1	ابا	AbA	11	2
2	ابابا	AbAbA	3	3
3	اباتت	AbAtt	4	3
4	اباتمتم	AbAtmtm	5	4
5	اباتنتن	AbAtntn	6	4
6	اباث	AbAv	8	3
7	اباثن	AbAvn	8	4
8	اباج	AbAj	11	3
9	اباجا	AbAjA	10	4
10	اباجن	AbAjn	11	4
11	باجوا	bAjwA	12	4

Table 3: An example illustrating how the next entry bit-skip and shared letter fields are used.

The first 12 bits store the distance in bits to the next test entry. If the word being searched for in the dictionary does not have a partial match with the test word at the current entry, the bit-skip field points to the next entry that does not begin with all the same letters. If a partial match is found, then only words between the current position and the bit-skip position may match the lookup word.

The remaining 4 bits store information on the

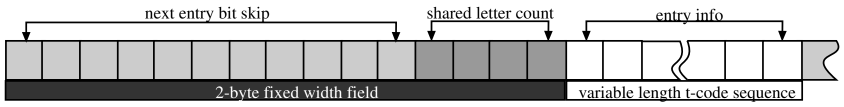


Figure 6: An entry using 12 bits for number of bits to skip to next entry and 4 bits for the number of shared letters.

number of letters shared between the current word and the next word. This allows the decoder to compare only the codes of the letters that have not been tested earlier, reducing the number of comparisons needed to make a match.

3.2.2 Results

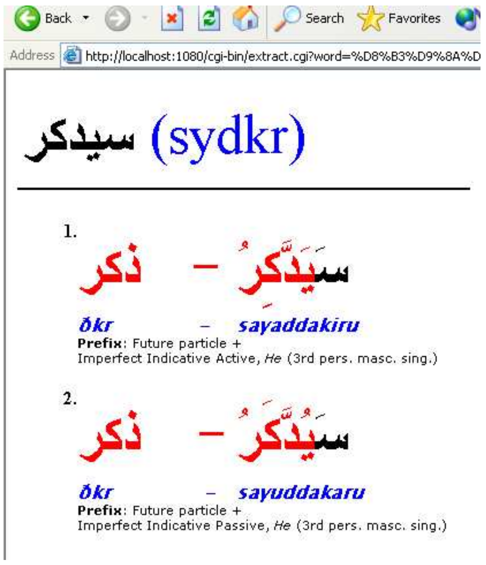
Using T-Codes and the indexing system described in this section, the dictionary disk-size was reduced to a mere 8 megabytes. The current dictionary size includes search and lookup information, which is over 90% smaller than the uncompressed B-Tree version with a comprable lookup speed.

Two devices may use a copy of the dictionary in order to communicate using T-Code transmission. A device may encode and transmit every Arabic word in a message into three codes containing the root, template, and affix identifiers for the word. The bandwidth used to transmit an Arabic word becomes a fraction of the equivalent T-Code encoded word.

For example, consider a word such as يَكتُبونَ (yaktubwna), which consists of the root, template, and affix identifier set {12884,460,30}. The T-Code lengths will depend only on the statistical frequency of each of these identifiers for all the words in an Arabic corpus so as to provide maximum efficiency; in this case the word may be represented as {0010101, 001001, 10100} and transmitted as 18 bits. Compare this size with the same word transmitted in Unicode. This 9 letter word would normally require 2,592 bits to be transmitted in raw Unicode(16-bit per character x 9 characters). If, instead, the raw identifier set was transmitted, it would require 48 bits (16-bit per integer x 3 integers), which is still significantly higher than the T-Code encoded form.

4 A Simple Application: A Root Extractor and Word Parser

To demonstrate the efficiency of the dictionary, we created a PERL based implementation of the decoder, and wrote a web CGI that derives and parses Arabic words. This particular implementation, although very simple, also functions as an accurate root extractor.



http://localhost:1080/cgi-bin/extract.cgi?word=%D8%B3%D9%8A%D

سيدكر (sydkr)

1.
ذكر – سَيَدَّكِرُ
ðkr – sayaddakiru
Prefix: Future particle +
Imperfect Indicative Active, *He* (3rd pers. masc. sing.)

2.
ذكر – سَيُدَّكَرُ
ðkr – sayuddakaru
Prefix: Future particle +
Imperfect Indicative Passive, *He* (3rd pers. masc. sing.)

Figure 7: Example output from the word-parser Web CGI using the T-Code encoded dictionary of Arabic words.

A UTF-8 Unicode-encoded HTML webpage accepts Arabic words in a simple form. The CGI is invoked with the input stripped of diacritics. Next, the CGI removes combinations of conjunction, prefix, and suffix letters that it finds in a pre-supplied list of affixes and it begins with the longest to the shortest sequences. The original word and each of its stripped forms are T-Code encoded and pushed into a queue. Entry codes that match any of the items in the queue are retrieved with their identifier lists from the dictionary and decoded. Identifiers are used in order to generate the words with diacritics that the entry identifies. Also, the identifier information is used to morphologically classify the entered word and the affixes that are used with it. The various possible morphological parsings are then output to HTML, as in Figure 7.

5 Further Work

We have described a system that uses T-Code to compress and access a comprehensive list of Arabic verbs by their morphological properties. Word generation here is restricted to verbs, but further research must extend the coverage to verbs and rootless words such as particles and loan words.

Once data has been obtained for word generation of nouns, the implementations of many of the applications discussed in the introduction become feasible. For example, a spell checker can be instructed to recognise conjunction, prefix, and suffix letter combinations, as described in Section 4. Since these letters do not cause alteration to adjacent letters, they may be removed and the remaining stem looked up in the dictionary. If a match is not found, a spelling error may be reported. Suggested spellings may come from the word-generation and transformation rules of the closest matching word or words. The closest match, like in English spell-checkers, would be the words that have reasonable character-correspondence.

Using the root-extraction algorithm in Section 4, root-based searching becomes possible. Both the search term and search text will undergo root extraction before a match is found.

Incremental searches such as that used in predictive text messaging only need to have a list of the conjunctions and affixes added to the dictionary list. The implementation can then allow combinations of conjunctions and affixes to attach to dictionary entries. Since the dictionary list now includes all forms affixed, transformed, and disguised, valid Arabic words will always find a match in the dictionary.

In the near future, we hope to increase the lookup and decoding speed by creating a T-Code Finite State Automaton (FSA) for the dictionary as described in (Nithyaganesh, 1998), which will be able to read an entire byte or two and output several code words. Currently, the decoding process tests if a code belongs to the T-Code set; if it does not match, another bit is added to the T-Code before it is tested once more. This continues until the code matches a code from the valid T-Code set. With a T-Code FSA, a significant improvement in the decoding speed will be witnessed, since bytes are looked up rather than bits.

References

Evan L Antworth. 1990. PC-KIMMO: a two-level processor for morphological analysis. *Occasional Publications in Academic Computing*, 16.

Kenneth R Beesley. 2001. Finite-state morphological analysis and generation of arabic at xerox research: Status and plans in 2001. In *ARABIC Language Processing: Status and Prospects*. Arabic NLP Workshop at ACL/EACL 2001, July.

The Unicode Consortium, 2003. *The Unicode Standard, Version 4.0*, chapter 2, page 29. Addison-Wesley, Reading, MA. ISBN 0-321-18578-1.

Jeffery E. F. Friedl. 2002. *Mastering Regular Expressions*. O'Reilly, 2nd edition, July.

Ulrich Gunther. 1998. *Robust Source Coding With Generalised T-Codes*. Ph.D. thesis, University of Auckland.

Lauri Karttunen and Kenneth R Beesley. 1992. Two-level rule compiler. Technical Report ISTL-92-2, Xerox, Xerox Palo Alto Research Center, Palo Alto, California.

L. Karttunen, J-P. Chanod, G. Grefenstette, and A. Schiller. 1997. Regular expressions for language engineering. In *Natural Language Engineering*, pages 238–305. February 5.

Lauri Karttunen. 1993. Finite-state lexicon compiler. Technical Report ISTL-NLTT-1993-04-02, Xerox, Xerox Palo Alto Research Center, Palo Alto, California, April.

Kirubalaratnam Nithyaganesh. 1998. *The Talk-Net Project Real-Time Speech Communication Using T-Codes*. MSc thesis, University of Auckland.

Mark R Titchener. 1984. Digital encoding by means of new t-codes to provide improved data synchronization and message integrity. In *Technical Note, IEE Proceedings*, volume 131 of *4*, pages 51–53, July.

Mark R Titchener. 1997. The synchronization of variable-length codes. *IEEE Transactions on Information Theory*, 43:683–691, March.

Performance Metrics for Word Sense Disambiguation

Trevor Cohn
Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
email: `tacohn@cs.mu.oz.au` fax: `+61-3-9348-1184`

Abstract

This paper presents the area under the Receiver Operating Characteristics (ROC) curve as an alternative metric for evaluating word sense disambiguation performance. The current metrics – accuracy, precision and recall – while suitable for two-way classification, are shown to be inadequate when disambiguating between three or more senses. Specifically, these measures do not facilitate comparison with baseline performance nor are they sensitive to non-uniform misclassification costs. Both of these issues can be addressed using ROC analysis.

1 Introduction

Word sense disambiguation (WSD) is one of the large open problems in the field of natural language processing, and in recent years has attracted considerable research interest (Ide and Veronis, 1998). The increasing availability of large corpora along with electronic sense inventories (such as WordNet; Fellbaum (1998)) has permitted the application of a raft of machine learning techniques to the task and provided an empirical means of performance evaluation. Until recently, most performance evaluation was conducted on disparate data sets, with only the *line* and *interest* corpora being used in a significant number of studies (Leacock et al., 1993; Bruce and Wiebe, 1994). SENSEVAL, a global evaluation performed in 1998 (Kilgarriff, 1998) and again in 2001 (Edmonds and Cotton, 2001), provided a common set of disambiguation tasks and performance evaluation criteria, allowing an objective comparison between competing methods.

These workshops included the tasks of disambiguating all words in a given text (the all-words task), and disambiguating each occurrence of a given word when it appears with a short context of a few surrounding sentences (the lexical sample task). Performance in the two tasks was measured in terms of precision and recall. Precision was defined as the proportion of classified instances that were correctly classified, and recall as the proportion of instances classified correctly – these allow for the possibility of an algorithm choosing not to classify a given instance. This evaluation criterion is insensitive to both the type of misclassification (is the predicted sense more closely related to the correct sense than other possible senses?) and the confidence with which the classifier has made the prediction (is the correct sense allocated a high probability despite not being given the highest value by the classifier?).

These problems led Resnik and Yarowsky (1999) to suggest an evaluation metric to provide partial credit for incorrectly classified instances. They penalise probability mass assigned to incorrect senses weighted by what they term the communicative/semantic distance between the that predicted sense and the correct sense. Using such measures, systems that confuse homographs would be penalised most heavily, while those that confuse fine-grained senses would only attract a minor penalty. The score assigned to a particular algorithm is highly reliant on the distances between senses; altering the relative penalties may well promote a previously non-optimal classifier to be the best performing classifier.

In order to highlight the problems in the existing evaluation methods, it is worth clarifying the qualities such a method should possess. Ideally, the evaluation metric should provide the following features:

(1) allow comparison of the performance of two or more classifiers on the same problem, ranking them in order of quality of prediction.

(2) penalise incorrectly classified instances based on the distance, or confusability between the predicted and correct sense, when disambiguating between three or more sentences. These penalties are henceforth referred to as (non-uniform) misclassification costs.

(3) allow comparison to baseline performance – that of the classifier which always predicts only the *a priori* majority sense.

(4) provide a readily interpretable measure of performance.

This paper analyses the metrics that have been used in assessing WSD performance in light of the above criteria. An alternative metric, Receiver Operating Characteristics (ROC), is proposed and shown to have favourable properties with respect to the criteria. Section 2 describes the shortcomings of the current metrics. Section 3 shows how ROC analysis can be applied to WSD evaluation. Section 4 provides a discussion in the context of empirical studies and I conclude in section 5 with thoughts for future study.

2 Problem Statement

Many comparisons of WSD performance use predictive accuracy as the sole means of comparison. Accuracy is defined as the proportion of instances that were disambiguated correctly, and is often compared to a baseline – the performance of the classifier that predicts the majority sense for every instance. Baseline performance varies greatly between words: from lower than 10% to greater than 90%. Without some form of normalisation, comparison of the results of different classifiers on different problems is impossible. The kappa statistic (Carletta, 1996) may be used to normalise accuracy, adjusting the result for the expected agreement with the perfect classifier by chance, thus satisfying criterion (3).

Implicit in the use of accuracy is the assumption that misclassification costs are equal (or equivalently, the set of senses are all equally similar to one another). Dictionary definitions and indeed, linguistic intuitions, tell us that some sense pairs are more closely related than others. A number of dictionaries present sense hierarchies for words based on their similarities. The guidelines used by lexicographers to determine what constitutes a homograph or sense vary considerably between dictionaries. Even individual lexicographers differ in their systematic preferences as to whether they conflate similar senses into one ('lumpers') or present them as a disparate set ('splitters') (Kilgarriff, 1997; Landau, 2001). Depending on the dictionary's purpose, factors such as frequency of occurrence, semantic and syntactic similarity, pronunciation and etymology of a given word are considered (with differing priority) when identifying word's senses. Accordingly, sense definitions are rarely compatible between different dictionaries (or thesauri), presenting issues for WSD tasks using only a single source as the sense inventory.

For a binary disambiguation task, misclassification costs should be uniform – we would not expect the cost of misclassifying an instance of sense$_a$ as sense$_b$ to be any different to the cost of misclassifying an instance of sense$_b$ as sense$_a$.[1] However, most words have many more than two senses; Zipf (1945) found the most commonly used words tend to have a much greater degree of polysemy than infrequently used words. While accuracy provides a good measure for comparison (satisfying criterion 1) and is simple to comprehend (4), it does not account for non-uniform classification costs (2), meaning that the ranking given will often not reflect the real costs of errors.

2.1 Precision and recall

These problems with accuracy led to the adoption of precision and recall instead of (or in addition to) accuracy for performance measurement. The combination of precision and recall have been used as the primary means of performance evaluation in the SENSEVAL exercises.

Precision and recall are commonly used metrics in information retrieval (IR) (Baeza-Yates and Ribeiro-Neto, 1999). The retrieval task often involves finding a small number of relevant documents from a large data repository. Algorithms are ranked based on their precision/recall tradeoff; an algorithm can be said to be better than another if it has higher precision (recall) for the

[1]This may not be true for all WSD tasks.

same or higher recall (precision). This provides only a loose ranking capacity (criterion 1).

Precision by itself is not a highly relevant measure in WSD as it focuses solely on the positive classifications, treating the negative instances as junk. Unlike IR classification, when disambiguating two senses of an ambiguous word, the set of positives is equally important as the set of negatives, since each corresponds to a distinct sense. The classification question could just as easily be phrased in the negative – this should not affect the performance measure. While high recall on its own would constitute a passable WSD method (in that the set of positive instances are largely correctly classified), high precision alone does not say much about the performance of the method. Simply selecting a single correct positive instance will yield the best possible precision, however, this method will perform woefully.[2] Similarly, classifying all instances as positive will achieve a recall of 1.0 and a precision of $\Pr(P)$ – the proportion of positive instances. As with predictive accuracy, the precision would need to be interpreted with respect to the baseline performance to allow comparisons between different tasks (hence having issues with criterion 3).

When extended to classification of three or more senses, these measures falter. In the case of SENSEVAL, the precision is redefined as the proportion of correctly predicted senses within the set of instances for which the algorithm hazarded a prediction, and recall as the proportion of correctly predicted senses over all instances. This implicitly allows classifiers to opt not to classify every instance. However non-exhaustive classifiers are of limited use, given that they must be combined with other classifiers in order to fully disambiguate a given text. Many tasks in which WSD forms a sub-task, such as machine translation (MT), require the word to be fully disambiguated – an unknown value is unacceptable.

Plotting the precision-recall curves (Manning and Schutze, 2000) allows for better performance ranking by optimising precision for a given level of recall. This goes some way in addressing the issues when assessing precision and recall with respect to criterion (1), however the problem exists as to what recall limit is acceptable – there is no theoretical justification for choosing a specific value, and modifying the value may well alter the rankings of the classifiers. The F-measure (a harmonic mean between precision and recall), may be used for simpler ranking providing a single number for comparison (4). However the weighting assigned to precision and recall in the calculation of the mean needs to be chosen and again, theory does not suggest what values to use.

Criterion (2) is not satisfied by this evaluation metric. The precision and recall values for disambiguation tasks involving three or more senses are based on the number of correct responses, ignoring the types of misclassification. Hence this method suffers for the same problems of predictive accuracy in this regard. Combining precision and recall measured for a number of binary disambiguation tasks for a single word (either between every pairing of senses or between each sense and all other senses) may go some way to satisfying (2) while remaining sensitive to the misclassification costs.

2.2 Semantic/communicative distance

Due to the insensitivity of accuracy and precision and recall to non-uniform misclassification costs, Resnik and Yarowsky (1999) proposed a metric incorporating the costs by weighting misclassification penalties by the distances between the predicted and correct senses. In such a manner misclassifications between fine-grained senses (eg., polysemy) will be penalised less harshly than those between coarser sense distinctions (eg., homonymy). They describe a sense hierarchy for the word *bank* derived from a single or multiple dictionaries, from which they derive a matrix of semantic distance between the senses.

The definition of a sense is a contentious issue within the field. The required granularity of sense distinctions varies with the task in which WSD is used. IR and speech synthesis require only coarse sense distinctions, however for MT and full text understanding much finer distinctions are required – often finer than offered by monolingual dictionaries. This would mean that the set of senses and the misclassification costs between senses, as approximated by the semantic distance, will be task dependent.

In most sense-tagged corpora, sense definitions

[2]Note also that selecting nothing will not yield a precision value at all, due to a division by zero.

have been taken from dictionary meanings or thesaurus categories. Granularity aside, these definitions have been criticised for the level of disagreement between lexicographers themselves (Kilgarriff, 1997). These result in markedly different descriptions of senses in different dictionaries, with no one dictionary offering a definitive set of sense description or more formal representation than all others. There is no reliable method of combining dictionary senses to reflect the level of granularity required by the task.

Resnik and Yarowsky went on to analyse the translation of different senses of a sample of ambiguous English words into 12 target languages. From this they estimated the probability of the senses being lexicalised differently in the translation into the target language. They found that between 52% (fine-grained polysemy) and 95% (homonymy) of senses were lexicalised differently on average in the target languages. They used these statistics to generate semantic distances between senses, reflecting the likelihood that the sense will have a different translation.

In such a scoring model the ranking of classifiers is highly sensitive to the sense hierarchy definition and its use in creating the distance matrix. If either of these were to change – and given the widespread disagreement between lexicographers with regard to sense definitions, this is highly possible – the set of classifiers would need to be reranked. Even when using the translation based measure of semantic distance, the use of a different set of target languages would be likely to affect the scoring. This has the potential to cause previously non-optimal classifiers to be re-ranked as optimal.

The semantic/communicative distance measure improves on the accuracy measure in that it accounts for non-uniform misclassification costs (2), while still providing a ranking measure (1). Translation based semantic distance measures sidestep a number of the issues involved with the use of dictionary sense inventories but are not without problems. The method still requires normalisation with the baseline performance (3), although the kappa statistic could also be used here. What is lost is simplicity (4) – the score assigned is not readily interpretable, as it is based on the distance matrix, an artificial construct based on unfounded assumptions.

3 ROC, an alternative metric

Receiver Operation Characteristic (ROC) graphs are an evaluation technique born in the field of signal detection which have become *de rigueur* in machine learning in recent years (Provost and Fawcett, 1997; Provost and Fawcett, 2001). A ROC graph plots the tradeoff between true positive rate and false negative rate in a binary classifier as a threshold value is modified. The true positive rate (TPR, or recall) is defined as the proportion of positive instances predicted as positive. The false positive rate (FPR, or fallout) is defined as the proportion of negative instances predicted as positive. The rationale behind graphing the relationship between these two factors for a given classifier is that various uses of the classifier may demand different optimisation criteria – such as maximising the TPR given a highest acceptable FPR, or finding the optimal classifier given the costs of errors and class distribution.

Provost and Fawcett described an algorithm for creating a ROC curve for a binary classifier and introduce the ROC convex hull (ROCCH), a method for determining the set of potentially optimal classifiers regardless of the misclassification costs and class distributions. Srinivasan (1999) extended ROC analysis to deal with non-binary classifiers, representing the rate by which each class is traded off for another class as each axis of ROC space. This leads to $c^2 - c$ dimensional ROC space, where c is the number of classes. The ROCCH can be calculated in $O(n^c)$ time, where n is the number of points in ROC space.

The sheer difficultly of visualising such high dimensional space prompted Fawcett to develop an alternative process. The area under the ROC curve (AUC) represents the probability that a binary classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This assigns a high score to those classifiers which form the majority of the ROCCH, or are consistently close to the hull. Fawcett (2001) extended AUC to cater for multiple classes by treating a c-dimensional classifier as c binary classifiers (each performing a one-vs-all classification), giving:

$$AUC_{total} = \sum_i AUC(c_i) \cdot \Pr(c_i)$$

where $\Pr(c_i)$ is the prior probability of the i-th sense.

WSD performance can be measured by the AUC metric, or by comparing a number of classifiers' performance curves in ROC space. Where the misclassification costs are known, the optimal classifier can be found simply by finding the point on the ROCCH with the lowest cost. The cost is simply the sum of the penalties assigned to incorrect classifications, which may be calculated from the semantic/communicative distances between senses as:

$$\sum_i \Pr(c_i) \sum_j r_{ij} d_{ij}$$

where r_{ij} is the proportion of instances of sense i classified as sense j, and d_{ij} is the distance between senses i and j, which is zero when $i = j$.

Where the misclassification costs are unknown or are not known precisely (as would be the case if Resnik and Yarowsky's was supplemented with confidence ranges for each cost), the ROCCH allows performance comparison between the different classifiers. The optimal sub-surface of the ROCCH can be found using the misclassification cost ranges meaning that only classifiers forming part of this sub-surface can be optimal. When the sub-surface is sufficiently small (i.e. the misclassification costs are known to a high degree of confidence) this should provide a good ranking of classifiers, as only a small number will form part of the optimal surface. This allows optimisation of learning methods that cannot incorporate non-uniform misclassification costs, as well as allowing optimisation where these costs are only known approximately and thus cannot be easily incorporated into classifier training. Storing the ROCCH allows this approach to be repeated if misclassification costs were to change.

When the sub-surface is quite large (i.e. when misclassification costs are not known precisely), it is likely that a number of classifiers will lie on the optimal surface. The AUC could then be used to discriminate between these classifiers, ranking those classifiers which are consistently closer to the ROCCH higher than those which are not. While the AUC doesn't strictly indicate optimality, it does provide a reasonable approximation.

This method allows comparison and loose ranking of classifiers (criterion 1), in that a number of classifiers can be discarded. Given precise misclassification costs (2), the classifiers (and indeed combinations of classifiers) can be readily ranked. The baseline performance is implicitly used in the analysis: only those classifiers which achieve better results than (weighted) random combinations of the trivial classifiers will be considered (3). This method has the added benefit of being robust in the face of changing or imprecise misclassification costs. While it does not provide a readily interpretable measure (4), especially when considering the convex hull in high dimensional space, the AUC can provide such a measure.

4 Empirical results and discussion

I have implemented three supervised WSD methods and analysed their performance using the three measures described above. All development was performed in the Natural Language Toolkit (Loper and Bird, 2002) and the source code is available as part of the toolkit. I implemented Yarowsky's (1994) decision list method, which he used for accent restoration in French and Spanish text (roughly similar to homograph disambiguation). This method uses the single most reliable piece of evidence in predicting the sense. I also implemented Brown et al.'s (1991) method, which was used for MT between French and English using decision trees to resolve the correct translation of each ambiguous word. Training uses the *flip-flop* algorithm (Nadas et al., 1991) to determine which feature will maximise the mutual information between a binary division of the values for that feature and the set of most probable senses given the feature takes one of those values. Both of these methods used collocates in a small window around the word as features. Lastly, I created a naive Bayes classifier (Manning and Schutze, 2000), using the unordered bag of words around the ambiguous word as the feature space. Words occurring fewer than five times in the corpora were ignored.

The three algorithms were compared on the interest corpus (Bruce and Wiebe, 1994). The word *interest* has six senses in the corpus with differing degrees of similarity to each other. Four experiments were performed; the first involved disam-

sense of *interest*	f	test$_1$	test$_2$	test$_3$	test$_4$
give attention	15%	✓	✓		✓
worthy of attention	1%	✓			✓
receiving attention	3%				✓
advantage	8%		✓		✓
share of company	21%			✓	✓
money	53%			✓	✓
baseline		97%	85%	72%	52%

Table 1: Test descriptions and baselines.

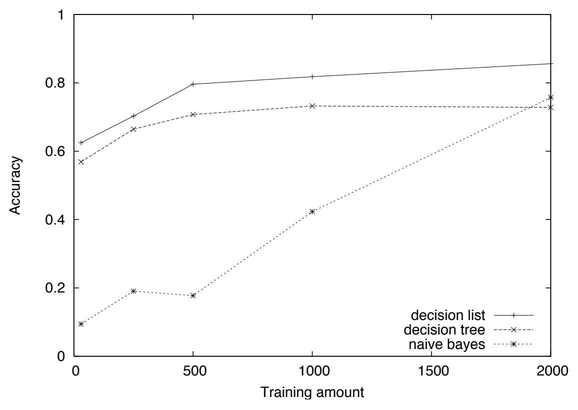


Figure 1: Learning curves

biguating between a pair of fine senses, which reported were difficult for human annotators (Bruce and Wiebe, 1998), and the second and third involve pairs of more distinct senses. The last test involved disambiguating between all six senses. Table 1 shows the gloss for each sense and the senses used for each test.

The learning curve, show in Figure 1, was constructed (in the same vein as Mooney's (1996) performance survey), showing the accuracy of each method on test 4 when trained with increasing amounts of data. It shows all three methods improving, with only the decision tree method showing signs of over-fitting. The accuracy, precision, recall and AUC values were measured and are shown in Table 2. Each test was performed using 10-fold cross validation. The precision, recall and AUC values were calculated with respect to the minority sense for tests 1 - 3. In test 4 both precision and recall are equal to the accuracy, as all three classifiers predict a sense for every instance. ROC curves were generated by ranking each instance (and predicted classification) in order of confidence, using the method described by Provost and Fawcett (2001), from which the AUC measures were calculated. The ROC curves for tests 1 - 3 are shown in Figure 2.

The decision list classifier is shown to be significantly more accurate than the other classifiers, exceeding the baselines for all tests, and performing extremely well for test 3. The results for test 1 are interesting in that the decision list method manages to outperform the baseline performance of 97%. With so few instances no solid conclusions may be drawn, however, the high AUC for the decision tree method suggests that it would perform better (in terms of predictive accuracy) by adjusting its threshold. This would allow it to operate at a more suitable point on its ROC curve, rather than at the origin.

The increase in performance of all methods from test 2 to 3 is most likely due to the increase in data. There are roughly three times as many instances in test 3, providing more training examples. Otherwise, the problems are quite similar, with similar ratios between the two senses. The AUC values support these conclusions, with the decision list and decision tree consistently outperforming naive Bayes for the first three tests. This can also be seen in the ROC curves (Figure 2), where these two classifiers largely dominate naive Bayes. Naive Bayes has a quite low AUC on all of the tests, while still being greater than the benchmark of 0.5. This is reflected in its lower accuracy in each test, however, in test 4, it outperforms the decision tree method despite having a much lower AUC. This suggests that the naive Bayes classifier is operating closer to the point which maximises accuracy on its ROC surface, whereas the decision tree is not. As earlier, this result suggests that the decision tree classifier should be operating with a lower threshold to achieve a higher accuracy. This is also evident in Figure 2, where the curve for the decision tree method, while largely dominated by the decision list curve, is still quite close to the ROCCH.

The highest accuracy classifier would fall on the ROC convex hull at a very steep gradient, due to the minority sense being treated as positive ($m = \frac{TPR}{FPR} = \frac{\Pr(s_b)}{\Pr(s_a)}$ where s_a and s_b are the minority and majority senses respectively). If misclassification costs were biased in favour of the minority sense, the difference in performance between the decision list and decision tree methods would be likely to be reduced, as can be seen from

	$test_1$	$test_2$	$test_3$	$test_4$
DL - accuracy	97.8	89.1	96.4	85.7
precision	0.8	31.1	26.5	85.7
recall	27.8	83.5	89.1	85.7
AUC	78.1	91.9	95.1	95.6
DT - accuracy	97.0	85.2	95.1	72.0
precision	0.0	27.9	25.4	72.0
recall	0.0	72.8	84.1	72.0
AUC	89.3	83.7	88.5	91.1
NB - accuracy	65.6	78.1	94.3	76.2
precision	3.3	37.1	26.3	76.2
recall	83.3	89.0	86.8	76.2
AUC	53.1	67.4	67.6	60.0

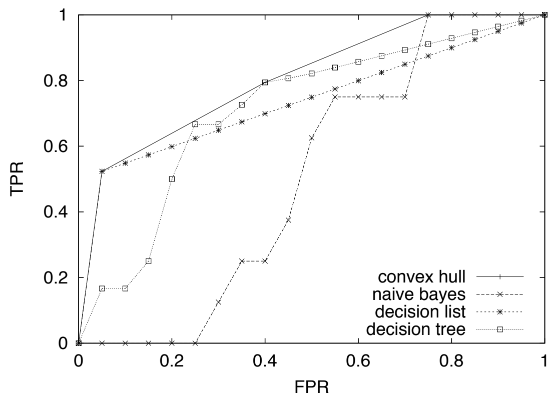
Table 2: Results expressed as percentages.

the proximity of their ROC curves at low gradients. The decision list classifier is shown to be superior to the other two, with higher AUC values on most tests and can be seen to be largely dominating the ROCCH for test 2 and test 3. If the misclassification costs are known at the time of training, a number of learning methods (i.e. naive Bayes) can incorporate them into the training phase, optimising the classifier with respect to these costs. However, this is not possible for all classifiers, requiring the use of ROC analysis to select the optimal classifier.

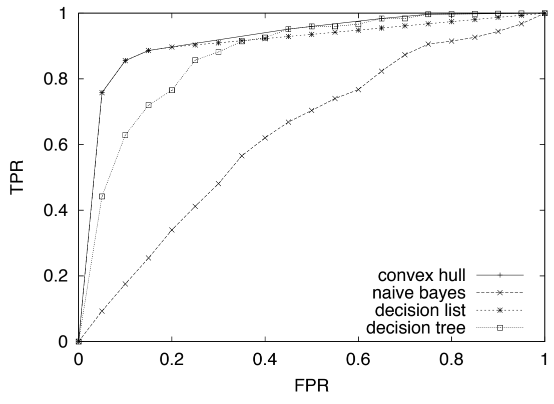
While the accuracy, precision and recall measures are relatively useful for analysing tests 1 - 3 (assuming uniform misclassification costs), they are not very useful in $test_4$. The manner in which they aggregate the set of incorrect classifications together loses a great deal of information about the classifier performance. The additional effort required in performing ROC analysis is well rewarded, with much more informative measures of performance.

5 Conclusion

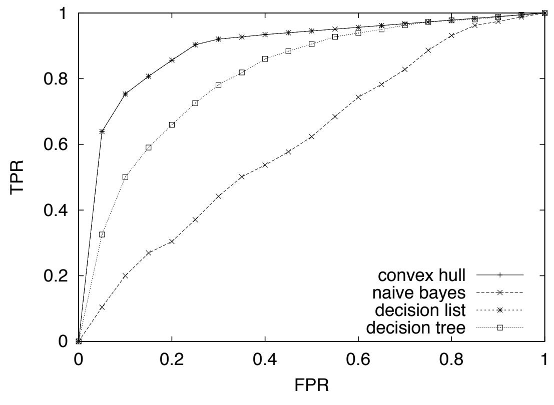
The nebulous nature of the word sense along with differing lexicographic practices mean that the task of WSD is ill-defined. Both dictionary and corpus based definitions of word senses, while not always agreeing on sets of senses for a given word, do concur that some sense pairs are more closely related than others. These relationships have been quantified in deriving the semantic/communicative distance matrix.



(a) Test 1



(b) Test 2



(c) Test 3

Figure 2: ROC curves for tests 1 – 3

ROC analysis proves to be a viable method for analysing performance, addressing a number of shortcomings with the existing measures. It has been shown to be of particular value in measuring performance when disambiguating between three or more senses. It satisfies the objectives of ease of comparison (1), taking misclassification costs into account (2) and implicitly incorporates baseline performance (3), while providing a simple and understandable measure (4) through the AUC. It has the added benefit of being flexible in the face of changing or imprecise misclassification costs. This is of particular significance in WSD given the vigour of the debate over what constitutes a sense, and as to how senses relate to each other. However, ROC analysis suffers from complexity in the form of high dimensional ROC space and computational demands in finding the convex hull.

SENSEVAL, and indeed the whole WSD field, stand to benefit from using ROC analysis as a performance metric. Further research into ROC analysis and its application to WSD and other natural language processing tasks can only help the field mature.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1991. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th Meeting of the Association for Computational Linguistics*, pages 264–270.

Rebecca Bruce and Janyce Wiebe. 1994. Word sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–145, Las Cruces, US.

Rebecca Bruce and Janyce Wiebe. 1998. Word sense distinguishability and inter-coder agreement. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP-98)*, Granada, Spain, June. Association for Computational Linguistics.

Jean Carletta. 1996. Assessing agreement on classifi cation tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: An overview. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5.

Tom Fawcett. 2001. Using rule sets to maximize ROC performance. In *2001 IEEE International Conference on Data Mining*, pages 131–138.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Nancy Ide and Jean Veronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):140.

Adam Kilgarriff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113.

Adam Kilgarriff. 1998. SENSEVAL: An exercise in evaluating word sense disambiguation programs. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 581–588, Granada, Spain.

Sidney I. Landau. 2001. *Dictionaries: The Art and Craft of Lexicography*. Cambridge University Press, second edition.

Claudia Leacock, Geoffrey Towell, and Ellen Voorhees. 1993. Towards building contextual representations of word senses using statistical models. In *SIGLEX workshop: Acquisition of Lexical Knowledge from Text, ACL*.

Edward Loper and Steven Bird. 2002. NLTK: The natural language toolkit. In *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, July. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schutze. 2000. *Foundations of Statistical Natural Language Processing*. MIT Press.

Raymond J. Mooney. 1996. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 82–91. Association for Computational Linguistics, Somerset, New Jersey.

Arthur Nadas, David Nahamoo, Michael A. Picheny, and Jeffrey Powell. 1991. An iterative flip-flop approximation of the most informative split in the construction of decision trees. In *International Conference on Acoustics, Speech, and Signal Processing*, New York.

Foster J. Provost and Tom Fawcett. 1997. Analysis and visualization of classifi er performance: Comparison under imprecise class and cost distributions. In *Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48.

Foster J. Provost and Tom Fawcett. 2001. Robust classifi cation for imprecise environments. *Machine Learning*, 42(3):203–231.

Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(2):113–134.

Ashwin Srinivasan. 1999. Note on the location of optimal classifi ers in n-dimensional ROC space. Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, Oxford.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95.

George Zipf. 1945. The meaning-frequency relationship of words. In *Journal of General Psychology*, volume 3, pages 251–256.

Straight to the Point: Discovering Themes for Summary Generation

Stephen Wan
Centre for Language Technology
Macquarie University
Sydney, Australia
swan@ics.mq.edu.au

Robert Dale
Centre for Language Technology
Macquarie University
Sydney, Australia
rdale@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, Australia
madras@ics.mq.edu.au

Cécile Paris
CSIRO Mathematical and Information Sciences
Locked Bag 17
North Ryde 1670
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

This paper[1] presents our approach to the problem of single sentence summarisation. We investigate the use of Singular Value Decomposition (SVD) to guide the generation of a summary towards the theme that is the focus of the document to be summarised. In doing so, the intuition is that the generated summary will more accurately reflect the content of the source document. Currently, we operate in the news domain and at present, our summaries are modelled on headlines. This paper presents SVD as an alternative method to determine if a word is a suitable candidate for inclusion in the headline. The results of a recall based evaluation comparing three different strategies to word selection, indicate that thematic information does help improve recall.

1 Introduction

In the midst of a plethora of archived electronic documents, the successful completion of a research task is affected by the ease with which users can quickly identify the relevant documents that satisfy their information needs. To do so, a researcher often relies on generated summaries that reflect the contents of the original document.

We explore the problem of generating a single sentence summary in the context of single document summarisation. Instead of identifying and extracting the most important sentence, we generate a new sentence from scratch. Thus, the resulting sentence summary may not occur verbatim in the source.

As a precursor to single sentence summarisation, we first explore the particular case of headline generation in the news domain, specifically English news. Our system re-uses words from the news article to generate a single sentence summary that resembles a headline. This is done by iteratively selecting a word from the source article and then appending it to previously selected words. This approach has been explored by a number of researchers (eg. see Witbrock and Mittal, 1999; Jin and Hauptmann, 2002). In existing approaches, a word is selected on the basis of two criteria: how well it acts as a summary word, and how grammatical it will be given the preceding summary words that have already been chosen. Our approach uses Singular Value Decomposition (SVD) in the first criterion, as a means of determining if a word is a good candidate for inclusion in the headline.

In this paper, we present an overview of our basic summarisation algorithm in Section 2. Section 3 examines limitations of the basic algorithm, illustrating how words can be used out of context, resulting in factually incorrect statements. This is the motivation of our SVD extension which is introduced conceptually in Section 4. Section 5 describes how we generate sentence summaries using SVD. In Section 6, we present our experimental design in which we

[1] This work was jointly funded by Centre for Language Technology at Macquarie University and CSIRO Mathematical and Information Sciences.

evaluated our approach, along with the results and corresponding discussion. Section 7, provides an overview of related work. Finally, in Section 8, we present our conclusions and future work.

2 Searching for a Probable Headline

We re-implemented the work described in Witbrock and Mittal (1999) to provide a single sentence summarisation mechanism. For full details of their approach, we direct the reader to their paper (Witbrock and Mittal, 1999). For an overview of our implementation of their algorithm, see Wan et al. (2003). For convenience, a brief description is presented here.

In a search, *n* words are selected on the basis of the two criteria. Conceptually, the task is twofold. Witbrock and Mittal (1999) label these two tasks as *Content Selection* and *Realisation*. Each criterion is scored probabilistically, whereby the probability is estimated by prior collection of corpus statistics.

To estimate Content Selection probability for each word, we use the Maximum Likelihood Estimate (MLE). In an offline training stage, the system counts the number of times a word is used in a headline, with the condition that it occurs in the corresponding news article. To form the probability, this frequency data is normalised by the number of times the word is used in articles across the whole corpus. This particular strategy of content selection, we refer to this as the Conditional probability. The Realisation criterion is determined simply by the use of bigram statistics, which provides an approximation of grammatical correctness when ordering selected words.

3 The Veracity of Generated Summaries

Berger and Mittal (2000) describe limitations of headlines generated by recycling words from the article. Differences in word order (for example, if the subject and object are reversed) can drastically affect sentence meaning.

However, we believe that the veracity of the generated sentence, with respect to the original document, is affected by a more basic problem than variation in word order. Because words from any part of a source document can be combined probabilistically, there is a possibility that words can be used together out of context. We refer to this as *Out-of-Context* error. Figure 1 presents an example of a generated headline in which the verb wrongly reports stock price movement. It also presents the actual context in which that verb was used.

Generated headline
"singapore stocks shares rebound"

Actual headline:
"Singapore shares fall, seen higher after holidays."

Original context of use of 'rebound':
"Singapore shares closed down below the 2,200 level on Tuesday but were expected to *rebound* immediately after Chinese Lunar New Year and Muslim Eid Al-Fitr holidays, dealers said."

Figure 1. An error in the generated headline due to a word being re-used out of context.

Out-of-Context errors arise due to limitations in the two criteria (presented in Section 1) for selecting words. Word selection is based on the previous usage of a word in headlines, not on its relevance to the current document being summarised. In addition, word order is modelled probabilistically using ngrams of lexemes. However, the semantic relationship implied by probabilistically placing two words next to each other, for example an adjective and a noun, might be suspect. As the name "Out-of-Context" suggests, this is especially true if the words were originally used in non-contiguous and unrelated contexts. This limitation in the word selection criteria can be characterized as being due to a lack of long distance relationship information.

4 Our Approach to "Encouraging Truth"

In response to this limitation, we explore the use of a matrix operation, *Singular Value Decomposition* (SVD) to guide the selection of words. Although our approach still does not guarantee factual correctness with respect to the source document, it has the potential to alleviate the Out-of-Context problem by improving the selection criteria for including words in the generated sentence, by considering the original contexts in which words were used. With this improved criteria, we hope to "encourage truth" by incorporating long distance relationships between words. Conceptually, SVD provides an analysis of the data which describes the relationship between the distribution of words and sentences. This analysis includes a grouping of sentences based on similar word distributions, which correspond to what we will refer to here as the main *themes* of the document.[2] By incorporating this information into the word selection criteria, the generated sentence will "gravitate" towards a single theme. That is, it will tend to use words from that theme, reducing the chance that words are placed together out of context.

Figure 2 presents an example of headlines generated with and without using SVD. The headline grammar is still problematic, however, in this example, the SVD headline is closer in meaning to the original headline. In contrast, the non-SVD headline uses words which are contentful but used out of context to form a non-meaningful string of words.

Actual Headline:
"China says conflict with U.S. unlikely."

First Sentence:
"Chinese Foreign Minister Qian Qichen said on Friday conflict between China and the United States was not possible unless Washington infringed on Beijing's sovereignty or territorial integrity."

Generated Headline without SVD:
"taiwan premier china visit rebel world war"

Generated Headline with SVD:
"china taiwan foreign minister said improved ties"

Figure 2. An example of headlines generated with and without SVD Content Selection.

By reflecting the content of the main theme, the summary may be *informative* (Borko, 1975). That is, the primary piece of information within the source document might be included within the summary. However, it would remiss of us to claim that this quality of the summary is guaranteed. In general, the generated summaries are at least useful to gauge what the source text is about, a characteristic described by Borko as being *indicative*.

[2] *Theme* is a term that is used in many ways by many researchers, and generally without any kind of formal definition. Our use of the term here is akin to the notion that underlies work on text segmentation, where sentences naturally cluster in terms of their 'aboutness'.

5 Using Singular Value Decomposition for Content Selection

As an alternative to the Conditional probability, we examine the use of SVD in determining the Content Selection probability. Before we outline the procedure for basing this probability on SVD, we will first outline our interpretation of the SVD analysis, based on that of Gong and Liu (2001). Our description is not intended to be a comprehensive explanation of SVD, and we direct the reader to Manning and Schütze (2000) for a description of how SVD is used in information retrieval.

Conceptually, when used to analyse documents, SVD can discover relationships between word co-occurrences in a corpus of text. For example, in the context of information retrieval, this provides one way to retrieve additional documents that contain synonyms of query terms, where synonymy is defined by similarity of word co-occurrences. By discovering patterns in word co-occurrences, SVD also provides information that can be used to cluster documents based on similarity of themes.

In the context of single document summarisation, we require SVD to cluster sentences based on similarities of themes. The SVD analysis provides information about how words and sentences relate to these themes. One such piece of information is a matrix of scores, indicating how representative the sentence is of each theme. Thus, for a sentence extraction summary, Gong and Liu (2001) would pick the top *n* themes, and for each of these themes, use this matrix to choose the sentence that best represents it.

For single sentence summarisation, we assume that the theme of the generated headline should match the most important theme of the article. The SVD analysis provides an ordering of themes, beginning with the one that accounts for the largest number of sentences, which we take to be the most important. The SVD analysis provides a matrix which scores how well each word relates to each theme. Given a theme, scores for each word, contained in a column vector of the matrix, can then be normalised to form a probability. The remainder of this section provides a more technical description of how this is done.

To begin with, we segment a text into sentences. Our sentence segmentation preprocessing is quite simple and based on the heuristics found in Manning and Schütze (2000). After removing stop words, we then form a terms (i.e. words) by sentences matrix, A. Each column of A represents a sentence. Each row represents the usage of a word in various sentences. Thus the frequency of word t in sentence s is stored in the cell A_{ts}. This gives us a $t * s$ matrix, where $t \geq s$. That is, we expect the lexicon size of a particular news article to exceed the number of sentences. For such a matrix, the SVD of A is a process that provides the right hand side of the following equation:

$$A = U.\Sigma.\, V^{\text{transpose}}$$

where U is a $t * r$ matrix, Σ is an $r * r$ matrix, and V is an $s * r$ matrix. The dimension size r is the rank of A, and is less than or equal to the number of columns of A, in this case, s. A diagram of this is presented in Figure 3.

It is important to note that the U matrix of the analysis provides information about how well words correspond to a particular theme. We examine the first column of the U matrix, sum the elements and then normalize each element by the sum to form a probability. This probability, which we refer to as the SVD probability, is then used as the Content Selection probability in the Viterbi search algorithm (Forney, 1973).

As an alternative to using the SVD probability and the Conditional Probability in isolation, a Combined Probability is calculated using the harmonic mean of the two. The harmonic mean was used in case the two component probabilities differed consistently in their respective orders of magnitude. Intuitively, when calculating a combined probability, this evens the importance of each component probability.

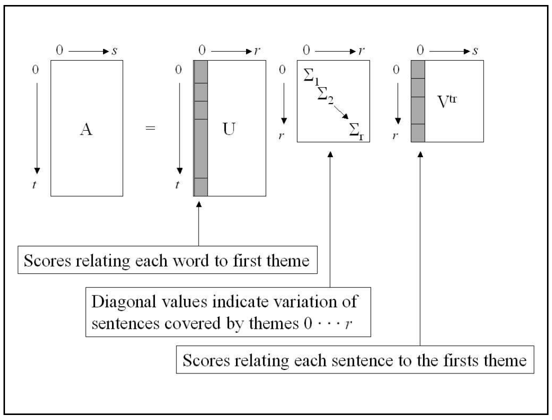

0 → s
0
t
A
=
0 → r
U
Σ1
Σ2
Σr
r
Vtr
Scores relating each word to first theme
Diagonal values indicate variation of sentences covered by themes 0 · · · r
Scores relating each sentence to the firsts theme

Figure 3. A diagram of our interpretation of the SVD matrices as it relates to single sentence summarisation.

To summarize, we end up with three alternative strategies in estimating the Content Selection Probability: the Conditional Probability, the SVD Probability and the Combined Probability.

6 Experiments

6.1 Data

In our experiments, we attempted to match the experimental conditions of Witbrock and Mittal (1999). We used news articles from the first six months of the Reuters 1997 corpus (Jan 1997 to June 1997). Specifically, we only examined news articles from the general Reuters category (*GCAT*) which covers primarily politics, sport and economics. This category was chosen not because of any particular domain coverage but because other categories exhibited frequent use of tabular presentation. The GCAT category contains in excess of 65,000 articles. Following Witbrock and Mittal (1999), we randomly selected 25,000 articles for training and a further 1000 articles for testing, ensuring that there was no overlap between the two data sets. During the training stage, we collected bigrams from the headline data, and the frequency of words occurring in headlines.

6.2 Experiment Design

We conducted an evaluation experiment to compare the performance of the three Content Selection strategies that we identified in Section 5: the Conditional probability, the SVD probability, and the Combined probability. We measure performance in terms of recall, i.e. how many of the words in the actual headline match words in the generated headline.[3] The recall metric is normalised to form a percentage by dividing the word overlap by the number of words in the actual headline.

For each test article, we generated headlines using each of the three strategies. For each strategy, we generated headlines of varying lengths, ranging from length 1 to 13, where the latter is the length of the longest headline found in the test set. We then compared the different strategies for generated headlines of equal length.

To determine if differences in recall scores were significant, we used the Wilcoxon Matched Pairs Signed Ranks (WMPSR) test (Seigel and Castellan, 1988). In our case, for a particular pair of Content Selection strategies, the alternate hypothesis was that the choice of Content Selection strategy affects recall performance. The null hypothesis held that there was no difference between the two content selection strategies. Our use of the non-parametric test was motivated by the observation that recall scores were not normally distributed. In fact, our results showed a positive skew for recall scores. To begin with, we compared the recall scores of the SVD strategy and the Conditional strategy in one evaluation. The strategy that was found to perform better was then compared with the Combined strategy.

In addition to the recall tests, we conducted an analysis to determine the extent to which the SVD strategy and the Conditional probability strategy were in agreement about which words to select for inclusion in the generated headline. For this analysis, we ignored the bigram probability of the Realisation component and just measured the agreement between the top *n* ranking words selected by each content selection strategy. Over the test set, we counted how many words were selected by both strategies, just one strategy, and no strategies. By normalising scores by the number of test cases, we determine the average agreement across the test set. We ran this experiment for a range of different values of N, ranging from 1 to 13, the length of the longest headline in the test set.

[3] Word overlap, whilst the easiest way to evaluate the summaries quantitatively, is an imprecise measure and must be interpreted with the knowledge that non-recall words in the generated headline might still indicate clearly what the source document is about.

6.3 Results

6.3.1 Recall Comparison

The results for the comparison of recall scores are presented in Table 1 and Table 2. Table 1 shows results of the WMPSR test when comparing the SVD strategy with the Conditional strategy.[4] Since the Conditional strategy was found to perform better, we then compared this with the Combined strategy, as shown in Table 2. From Table 1, it is clear that, for all sentence lengths, there is a significant difference between the SVD strategy and the Conditional strategy, and so we reject the null hypothesis. Similarly, Table 2 shows that there is a significant difference between the Conditional strategy and the Combined strategy, and again we reject the null hypothesis. We conclude that SVD probability alone is outperformed by the Conditional probability; however, using both probabilities together leads to a better performance.

6.3.2 Agreement between Strategies

The agreement between strategies is presented in Table 3. Interestingly, of the words recalled, the majority have only been selected by one content selection strategy. That is, the set of words recalled by one content selection strategy do not necessarily subsume the set recalled by the other. This supports the results obtained in the recall comparison in which a combined strategy leads to higher recall. Interestingly, the last column in the table shows that the potential combined recall is greater than the recall achieved by the combined strategy; we will return to this point in Section 6.4.

[4] The performance of our Conditional strategy is roughly comparable to the results obtained by Banko, Mittal and Witbrock (2000), in which they report recall scores between 20% to 25%, depending on the length of the generated headline.

Sentence Length	Average Recall : SVD	Average Recall : Cond.	Probability	Reject H_0
1	03.68%	03.98%	$p \leq 0.0$	yes
2	07.02%	06.97%	$p \leq 0.5$	yes
3	10.05%	11.44%	$p \leq 0.0$	yes
4	12.39%	13.90%	$p \leq 0.0$	yes
5	14.21%	15.73%	$p \leq 0.0$	yes
6	15.57%	17.84%	$p \leq 1.1e-05$	yes
7	16.59%	19.14%	$p \leq 1.8e-07$	yes
8	17.74%	20.30%	$p \leq 1.3e-07$	yes
9	18.74%	21.33%	$p \leq 1.3e-06$	yes
10	19.73%	22.44%	$p \leq 1.0e-06$	yes
11	20.19%	23.50%	$p \leq 2.2e-10$	yes
12	20.85%	24.54%	$p \leq 4.4e-13$	yes
13	21.13%	25.13%	$p \leq 1.4e-12$	yes

Table 1. A comparison of recall scores for the SVD strategy and the Conditional strategy.

Sentence Length	Average Recall : Cond	Average Recall : Combined	Probability	Reject H_0
1	03.98%	04.05%	$p \leq 0.1305$	yes
2	06.97%	08.60%	$p \leq 2.8e-13$	yes
3	11.44%	12.34%	$p \leq 0.0007$	yes
4	13.90%	15.44%	$p \leq 8.5e-09$	yes
5	15.73%	17.33%	$p \leq 1.9e-09$	yes
6	17.84%	18.72%	$p \leq 0.0003$	yes
7	19.14%	20.34%	$p \leq 1.3e-05$	yes
8	20.30%	21.48%	$p \leq 2.9e-06$	yes
9	21.33%	22.60%	$p \leq 4.0e-06$	yes
10	22.44%	23.82%	$p \leq 1.2e-06$	yes
11	23.50%	24.56%	$p \leq 0.0003$	yes
12	24.54%	25.44%	$p \leq 0.0008$	yes
13	25.13%	26.37%	$p \leq 8.6e-06$	yes

Table 2. A comparison of recall scores for the Conditional strategy and the Combined strategy.

Sentence Length	Selected by neither method	Selected by only 1 method	Selected by both methods	Total Recall
1	91.6%	8.0%	0.3%	8.3%
2	84.7%	14.1%	1.0%	15.1%
3	79.9%	17.5%	2.5%	20.0%
4	76.6%	19.3%	3.9%	23.2%
5	73.8%	21.0%	5.1%	26.1%
6	71.4%	22.1%	6.4%	28.5%
7	69.6%	22.4%	7.8%	30.2%
8	67.9%	22.9%	9.1%	32.0%
9	66.4%	23.2%	12.3%	35.5%
10	65.0%	23.5%	11.3%	34.8%
11	63.9%	23.6%	12.3%	35.9%
12	63.0%	23.6%	13.2%	36.8%
13	62.1%	23.5%	14.3%	37.8%

Table 3. Agreement of words chosen between the SVD strategy and the Conditional probability strategy to content selection

6.4 Discussion

The SVD strategy ultimately did not perform as well as we might have hoped. There are a number of possible reasons for this.

1. Whilst using the Combined probability did lead to a significantly improved result, this increase in recall was only small. Indeed, the analysis of the agreement between the Conditional strategy and the SVD strategy indicates that the current method of combining the two probabilities is not optimal and that there is still considerable margin for improvement.

2. Even though the recall of the SVD strategy was poorer by a only a few percent, the lack of improvement in recall is perplexing, given that we expected the thematic information to ensure words were used in correct contexts. There are several possible explanations, each warranting further investigation. It may be the case that the themes identified by the SVD analysis were quite narrow, each encompassing only a small number of sentences. If so, certain words occurring in sentences outside the theme would be given a lower probability even if they were good headline word candidates. Further investigation is necessary to determine if this is a shortcoming of our SVD strategy or an artefact of the domain. For example, it might be the case that the sentences of news articles are already thematically quite dissimilar.

3. One might also question our experimental design. Perhaps the kind of improvement brought about when using the SVD probability cannot be measured by simply counting recall. Instead, it may be the case that an evaluation involving a panel of judges is required to determine if the generated text is qualitatively better in terms of how faithful the summary is to the information in the source document. For example, a summary that is more accurate may not necessarily result in better recall. Finally, it is conceivable that the SVD strategy might be more sensitive to preprocessing stages such as sentence delimitation and stopword lists, which are not necessary when using the Conditional strategy.

Despite these outstanding questions, there are pragmatic benefits when using SVD. The conditional strategy requires a paired training set of summaries and source documents. In our case, this was easily obtained by using headlines in lieu of single sentence summaries. However, in cases where a paired corpus is not available for training, the SVD strategy might be more appropriate, given that the performance does not differ considerably. In such a situation, a collection of documents is only necessary for collecting bigram statistics.

7 Related Work

As the focus of this paper is on statistical single-sentence summarisation we will not focus on preceding work which generates summaries greater in length than a sentence. We direct the reader to Paice (1990) for an overview of summarisation based on sentence extraction. Examples of recent systems include Kupiec et al. (1995) and Brandow et al. (1995). For examples of work in producing abstract-like summaries, see Radev and McKeown (1998), which combines work in information extraction and natural language processing. Hybrid methods for abstract-like summarisation, which combine statistical and symbolic approaches, have also been explored; see, for example, McKeown et al. (1999), Jing and McKeown (1999), and Hovy and Lin (1997).

Statistical single sentence summarisation has been explored by a number of researchers (see for example, Witbrock and Mittal, 1999; Zajic et al., 2002). Interestingly, in the work of Witbrock and Mittal (1999), the selection of words for inclusion in the headline is decided solely on the basis of corpus statistics and does not use statistical information about the distribution of words in the document itself. Our work differs in that we utilise an SVD analysis to provide information about the document to be summarized, specifically its main theme.

Discourse segmentation for sentence extraction summarisation has been studied in work such as Boguraev and Neff (2000) and Gong and Liu (2001). The motivation behind discovering segments in a text is that a sentence extraction summary should choose the most representative sentence for each segment, resulting in a comprehensive summary. In the view of Gong and Liu (2001), segments form the main themes of a document. They present a theme interpretation of the SVD analysis, as it is used for discourse segmentation, upon which our use of the technique is based. However, Gong and Liu use SVD for creating sentence extraction summaries, not for generating a single sentence summary by re-using words.

In subsequent work to Witbrock and Mittal (1999), Banko et al. (2000) describe the use of information about the position of words within four quarters of the source document. The headline candidacy score of a word is weighted by its position in one of the quarters. We interpret this use of position information as a means of guiding the generation of a headline towards the central theme of the document, which for news articles typically occurs in the first quarter. SVD potentially offers a more general mechanism for handling the discovery of the central themes and their positions within the document.

Jin et al. (2002) have also examined a statistical model for headlines in the context of an information retrieval application. Jin and Hauptmann (2001) provide a comparison of a variety of learning approaches used by researchers for modelling the content of headlines including the Iterative Expectation-Maximisation approach, the K-Nearest neighbours approach, a term vector approach and the approach of Witbrock and Mittal (1999). In this comparison, the approach of Witbrock and Mittal (1999) fares favourably, ranking second after the term vector approach to title word retrieval (see Jin and Hauptmann, 2001, for details). However, while it performs well, the term vector approach Jin et al. (2002) advocate doesn't explicitly try to model the way a headline will usually discuss the main theme and may thus be subject to the Out-of-Context problem.

8 Conclusion

Combining both the SVD probability and Conditional probability marginally improves recall; lending support to the intuition that thematic information may help generate better single sentence summaries by avoiding out-of-context errors. However, there are still many unanswered questions. In future work, we intend to investigate these techniques in a domain other than news text so that we can draw conclusions as to how well these strategies generalise to other genres. We also intend to conduct user evaluations to gauge the quality of the generated summaries for both the Conditional and the SVD strategies. Finally, we are interested in how well this approach works with other languages. Preliminary results with Chinese headline generation are promising.

References

Banko M., Mittal V., and Witbrock M. (2000) Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Boguraev B., and Neff M. (2000) Discourse segmentation in aid of document summarization. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS- 33), Minitrack on Digital Documents Understanding*. Maui, Hawaii: IEEE.

Borko, H., and Bernier, C. (1975) *Abstracting Concepts and Methods*. New York: Academic Press.

Brandow, R., Mitze, K., and Rau, L. (1995) Automatic condensation of electronic publications by sentence selection. In *Information Processing and Management*, 31(5), pages 675-685.

Forney G. D. (1973) The Viterbi Algorithm. In the *Proceedings of the IEEE,* pages 268-278.

Gong Y., and Liu, X. (2001) Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In the *Proceedings SIGIR 2001*: pages 19-25.

Hovy, E. and Lin, C. (1997) Automated text summarization in SUMMARIST. In the *Proceedings of ACL-EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 18-24.

Jin, R., and Hauptmann, A. (2001) Learning to Select Good Title Words: An New Approach based on Reversed Information Retrieval. In the *Proceedings of the Eighteen International Conference on Machine Learning (ICML 2001),* Williams College,MA, June 28-July 1.

Jin, R., Zhai, C., and Hauptmann, A. (2002) Title language model for information retrieval. In the *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002),* Tampere, Finland, August 11-15.

Jing, H., and McKeown, K. (1999) The decomposition of human-written summary sentences. In the *Proceedings of the 22nd Conference on Research and Development in Information Retrieval (SIGIR--99).*

Kupiec, J., Pedersen, J., and Chen, F. (1995) A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Fox, E., Ingwersen, P., and Fidel, R. (Editors), pages 68—73.

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

Marcu, D. (2000) *The Theory and Practice of Discourse Parsing and Summarization*. Cambridge: The MIT Press.

McKeown, K., Klavans, J., Hatzivassiloglou, V., Barzilay, R., and Eskin, E. (1999) Towards multidocument summarization by reformulation: Progress and prospects. In the *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI--99).*

Paice, C. (1990) Constructing Literature Abstracts by Computers: Techniques and Prospects. In *Information Processing and Management*, Vol. 26, No. 1, pages 171–186.

Radev, D. and McKeown, K. (1998) Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469-500, September.

Siegel, Sidney and Castellan, Jr. N. John. (1988) *Nonparametric Statistics For The Behavioral Sciences*. McGraw-Hill, Inc., second edition.

Wan, S., Dras, M., Dale, R., and Paris, C. (2003) Using Thematic Information in Statistical Headline Generation. In the *Proceedings of the Multilingual Summarization and Question Answering Workshop*. Sapporo, Japan.

Witbrock, M., and Mittal, V. (1999) Ultrasummarization: A statistical approach to generating highly condensed non-extractive summaries. In the *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR '99).*

Zajic D., Door B., and Schwartz R. (2002) Automatic Headline Generation for Newspaper Stories. In the *Proceedings of the Document Understanding Conference (DUC 2002).*

Resolving Sense Ambiguity of Korean Nouns Based on Concept Co-occurrence Information

You-Jin Chung **Jong-Hyeok Lee**
Div. of Electrical and Computer Engineering
Pohang University of Science and Technology (POSTECH) and Advanced Information Technology Research Center(AlTrc)
San 31 Hyoja-dong, Nam-gu, Pohang 790-784, R. of Korea
prizer@postech.ac.kr jhlee@postech.ac.kr
fax: +82-54-279-5699 fax: +82-54-279-5699

Abstract

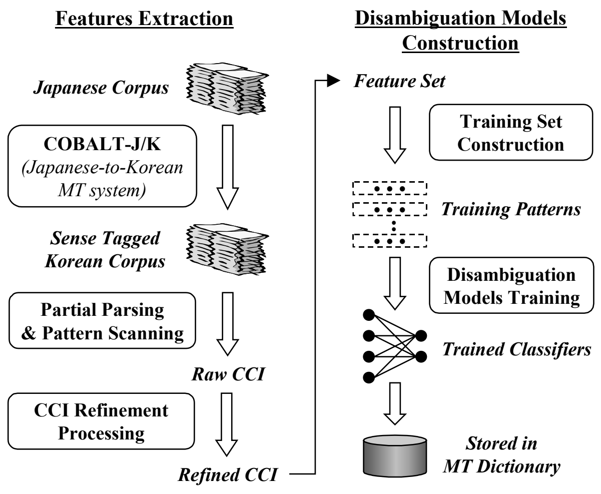
From the view point of the linguistic typology, Korean and Japanese have many grammatical similarities which enable it to easily construct a sense-tagged Korean corpus through an existing high-quality Japanese-to-Korean machine translation system. The sense-tagged corpus may serve as a knowledge source to extract useful clues for word sense disambiguation (WSD). This paper addresses a disambiguation model for Korean nouns, whose execution is based on the concept codes extracted from the sense-tagged corpus and the semantic similarity values over a thesaurus hierarchy. By the help of the automatically constructed sense-tagged corpus, we overcome the knowledge acquisition bottleneck. Also, we show that the performance of word sense disambiguation can be improved by combining several base classifiers. In an experimental evaluation, the proposed model using a majority voting achieved an average precision of 77.75% with an improvement over the baseline by 15.00%, which is very promising for real world MT systems.

1 Introduction

Generally, a Korean homograph may be translated into a different Japanese equivalent depending on which sense is used in a given context. Thus, noun sense disambiguation is essential to the selection of an appropriate Japanese target word in Korean-to-Japanese translation.

Much research on word sense disambiguation has revealed that several different types of information can contribute to the resolution of lexical ambiguity. These include surrounding words (an unordered set of words surrounding a target word), local collocations (a short sequence of words near a target word, taking word order into account), syntactic relations (selectional restrictions), parts of speech, morphological forms, semantic context, etc (McRoy, 1992, Yarowsky, 1992, Ng and Zelle, 1997).

To extract such information, various types of knowledge sources have been utilized such as machine-readable dictionaries (MRD), thesauri, and computational lexicons. Since most MRDs and thesauri were created for human use and display inconsistencies, these resources have clear limitations. Sense-tagged corpora have been used as the most useful knowledge source for WSD. However, despite the value of sense-tagged corpora, two major obstacles impede the acquisition of lexical knowledge from corpora: the difficulties of manually sense-tagging a training corpus, and data sparseness (Ide and Veronis, 1998). Manual sense-tagging of a corpus is extremely costly, and at present, very few sense-tagged corpora are available.



Features Extraction
Disambiguation Models Construction
Japanese Corpus
COBALT-J/K (Japanese-to-Korean MT system)
Sense Tagged Korean Corpus
Partial Parsing & Pattern Scanning
Raw CCI
CCI Refinement Processing
Refined CCI
Feature Set
Training Set Construction
Training Patterns
Disambiguation Models Training
Trained Classifiers
Stored in MT Dictionary

Figure 1. System Architecture

In our WSD approach, we construct a sense-tagged corpus automatically by using a method based on similarities between Korean and Japanese. Our disambiguation model is based on the work of Li *et al* (2000), especially focusing on the practicality of the method for application to real world MT systems. We alleviate the data sparseness problem by adopting a concept-based processing and reduce the number of features to a practical size by refinement processing.

This paper is organized as follows. Section 2 presents the overall system architecture. Section 3 explains the automatic construction of a sense-tagged Korean corpus and the extraction of refined features for word sense disambiguation. Section 4 describes the construction of feature set and the learning of disambiguation models. In Section 5, the experimental results are given, showing that the proposed method may be useful for WSD in a real text. In this paper, Yale Romanization is used to represent Korean expressions.

2 System Architecture

Our disambiguation method consists of two phases. The first phase is the extraction of features for WSD and the second phase is the construction of disambiguation models. (see Figure 1.)

For practical reasons, a reasonably small number of features is essential to the design of disambiguation models. To construct a feature set of a reasonable size, we adopt Li's method (2000), based on concept co-occurrence information (CCI). CCI are concept codes of words which co-occur

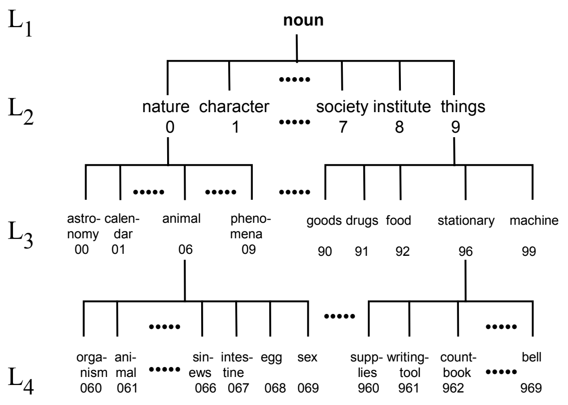


Figure 2. Concept hierarchy of the Kadokawa thesaurus

with the target word for a specific syntactic relation.

In accordance with Li's method, we automatically extract CCI from a corpus by constructing a sense-tagged Korean corpus. To accomplish this, we apply a Japanese-to-Korean MT system. Next, we extract CCI from the constructed corpus through partial parsing and scanning. To eliminate noise and to reduce the number of CCI, refinement processing is applied to the extracted raw CCI set. After completing refinement processing, we use the remaining CCI as features for disambiguation. The obtained feature set and the trained disambiguation models are stored in a dictionary for MT system.

3 Extraction of Features for WSD

3.1 Automatic Construction of Sense-tagged Corpus

Japanese and Korean are very similar in word order and lexical properties. Also, they have many nouns in common derived from Chinese characters. Because almost all Japanese common nouns represented by Chinese characters are monosemous, little transfer ambiguity is exhibited in Japanese-to-Korean translation of nouns, and we can obtain a sense-tagged Korean corpus of a good quality by using those linguistic similarities between Korean and Japanese.

For automatic construction of the sense-tagged corpus, we used a Japanese-to-Korean MT system

Table 1. Structure of CCI Patterns

CCI type	Structure of pattern
$type_0$	unordered co-occurrence words
$type_1$	**noun** + noun or noun + **noun**
$type_2$	noun + *uy* + **noun**
$type_3$	**noun** + other particles + noun
$type_4$	**noun** + *lo/ulo* + verb
$type_5$	**noun** + *ey* + verb
$type_6$	**noun** + *eygey* + verb
$type_7$	**noun** + *eyse* + verb
$type_8$	**noun** + *ul/lul* + verb
$type_9$	**noun** + *i/ka* + verb
$type_{10}$	verb + relativizer + **noun**

Table 2. Concept codes and frequencies in CFP ($\{<C_i,f_i>\}$, $type_2$, *nwun*(eye))

Code	Freq.	Code	Freq.	Code	Freq.	Code	Freq.
028	19	107	8	121	7	126	4
143	8	160	5	179	7	277	4
320	8	331	6	416	7	429	22
433	4	501	13	503	10	504	11
505	6	507	12	508	27	513	5
530	6	538	11	552	4	557	7
573	5	709	5	718	5	719	4
733	5	819	4	834	4	966	4
987	9	other*	210				

※ 'other' in the table means the set of concept codes with the frequencies less than 4.

called COBALT-J/K[1]. In the transfer dictionary of COBALT-J/K, nominal and verbal words are annotated with concept codes of the Kadokawa thesaurus (Ohno and Hamanishi, 1981), which has a 4-level hierarchy of about 1,100 semantic classes, as shown in Figure 2. Concept nodes in level L_1, L_2 and L_3 are further divided into 10 subclasses.

We made a slight modification of COBALT-J/K to enable it to produce Korean translations from a Japanese text, with all content words tagged with specific concept codes at level L_4 of the Kadokawa thesaurus. As a result, a sense-tagged Korean corpus of 1,060,000 sentences can be obtained from the Japanese corpus (*Asahi Shinbun*, Japanese Newspaper of Economics, etc.).

The quality of the constructed sense-tagged corpus is a critical issue. To evaluate the quality, we collected 1,658 sample sentences (29,420 eojeols[2]) from the corpus and checked their precision. The total number of errors was 789, and included such errors as morphological analysis, sense ambiguity resolution and unknown words. It corresponds to the accuracy of 97.3% (28,631 / 29,420 eojeols). The number of ambiguity resolution errors was 202 and it took only 0.69% of the overall corpus (202 / 29,420 eojeols). Considering the fact that the overall accuracy of the constructed corpus exceeds 97% and only a few sense ambiguity resolution errors were found in the Japanese-to-Korean translation of nouns, we regard the generated sense-tagged corpus as highly reliable.

3.2 Extraction of Raw CCI

Unlike English, Korean has almost no syntactic constraints on word order as long as the verb appears in the final position. The variable word order often results in discontinuous constituents. Instead of using local collocations by word order, Li *et al.* (2000) defined 13 patterns of CCI for homographs using syntactically related words in a sentence. Because we are concerned only with noun homographs, we adopt 11 patterns from them excluding verb patterns, as shown in Table 1. The words in bold indicate the target homograph and the words in italic indicate Korean particles.

For a homograph W, concept frequency patterns (CFPs), i.e., ($\{<C_1,f_1>,<C_2,f_2>, \ldots , <C_k,f_k>\}$, $type_i$, $W(S_i)$), are extracted from the sense-tagged training corpus for each CCI type i by partial parsing and pattern scanning, where k is the number of concept codes in $type_i$, f_i is the frequency of concept code C_i appearing in the corpus, $type_i$ is an CCI type i, and $W(S_i)$ is a homograph W with a sense S_i. All concepts in CFPs are three-digit concept codes at level L_4 in the Kadokawa thesaurus. Table 2 demonstrates an example of CFP that can co-occur with the homograph '*nwun*(eye)' in the form of the CCI $type_2$ and their frequencies.

3.3 CCI Refinement Processing

The extracted CCI set is too numerous and too noisy to be used in a practical system, and must to be further selected. To eliminate noise and to reduce the number of CCI to a practical size, we ap-

[1] COBALT-J/K (Collocation-Based Language Translator from Japanese to Korean) is a high-quality practical MT system developed by POSTECH.

[2] An Eojeol is a Korean syntactic unit consisting of a content word and one or more function words.

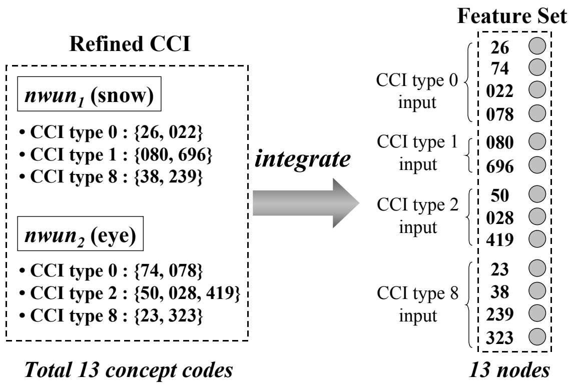


Figure 3. Construction of Feature Set for '*nwun*'

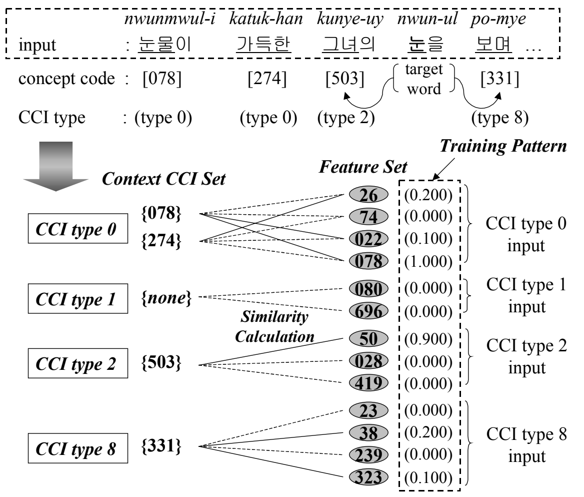


Figure 4. Construction of Training Pattern by Using Concept Similarity Calculation

ply the refinement processing to the extracted CCI set. CCI refinement processing is composed of 2 processes: concept code discrimination and concept code generalization.

3.3.1 Concept Code Discrimination

In the extracted CCI set, the same concept code may appear for determining the different meanings of a homograph. To select the most probable concept codes, which frequently co-occur with the target sense of a homograph, Li defined the discrimination value of a concept code using Shannon's entropy. A concept code with low entropy has a large discrimination value. If the discrimination value of the concept code is larger than a threshold, the concept code is selected as useful information for deciding the word sense. Otherwise, the concept code is discarded.

3.3.2 Concept Code Generalization

After concept discrimination, co-occurring concept codes in each CCI type must be further selected and the code generalized. To perform code generalization, Li adopted Smadja's work (Smadja, 1993) and defined the code strength using a code frequency and a standard deviation in each level of the concept hierarchy. The generalization filter selects the concept codes with a strength larger than a threshold. We perform this generalization processing on the Kadokawa thesaurus level L_4 and L_3.

After processing, the system stores the refined conceptual patterns ($\{C_1, C_2, C_3, ...\}$, $type_i$, $W(S_i)$) as a knowledge source for WSD of real texts. These refined CCI are used as features for disambiguation models. The more specific description of the CCI extraction is explained in Li (2000).

4 Construction of Disambiguation Models

4.1 Feature Set Construction

The feature set is constructed by integrating the extracted CCI into a single vector. Figure 3[3] demonstrates a construction example of the feature set for the homograph '*nwun*' with the sense 'snow' and 'eye'. The left side is the extracted CCI for each sense after refinement processing. We construct the feature set for '*nwun*' by merely integrating the concept codes in CCI set of both senses. The resulting feature set is partitioned into several subgroups depending on their CCI types, i.e., type 0, type 1, type 2 and type 8. Since the extracted CCI set are different according to each word, each homograph has a feature set of its own.

4.2 Extraction of Training Patterns

After constructing the feature set for WSD, we extract training patterns for each homograph from the previously constructed sense-tagged corpus. The construction of training pattern is performed in the following 2 steps.

Step 1. Extract CCI from the context of the target homograph. The window size of the context is

[3] The concept codes in Figure 3 are simplified ones for the ease of illustration. In reality there are 87 concept codes for '*nwun*'.

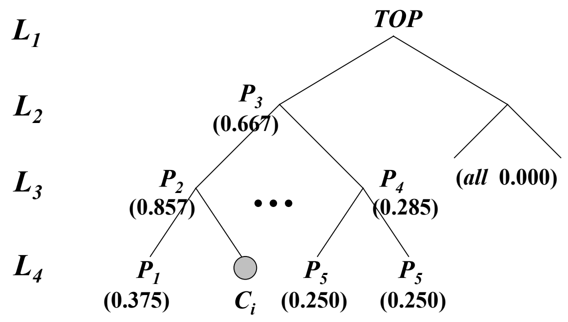


Figure 5. Concept Similarity on the Kadokawa Thesaurus Hierarchy

a single sentence. Consider, for example, the sentence in Figure 4 which has the meaning of "Seeing her eyes filled with tears, …". The target homograph is the word '*nwun*'. We extract its CCI from the sentence by partial parsing and pattern scanning. In Figure 4, the words '*nwun*' and '*kunye*(her)' with the concept code 503 have the relation of <noun + *uy* + noun>, which corresponds to '*CCI type 2*' in Table 1. There is no syntactic relation between the words *'nwun'* and '*nwun-mul*(tears)' with the concept code 078, so we assign '*CCI type 0'* to the concept code 078.

Similarly, we can obtain all pairs of CCI types and their concept codes appearing in the context. All the extracted <*CCI-type*: *concept codes*> pairs are as follows: {<*type 0*: *078,274*>, <*type 2*: *503*>, <*type 8*: *331*>}.

Step 2. Obtain the training pattern by calculating concept similarities between concept codes in the context CCI set and the feature set. Concept similarity calculation is performed only between the concept codes with the same CCI-type. This score represents that how much each node of the network relates to clues appearing in the target context. The calculated concept similarity score is assigned to each feature node as the activation strength for it.

Csim(C_i, P_j) in Equation 1 is used to calculate the concept similarity between C_i and P_j, where *MSCA*(C_i, P_j) is the most specific common ancestor of concept codes C_i and P_j, and *weight* is a weighting factor reflecting that C_i as a descendant of P_j is preferable to other cases. That is, if C_i is a descendant of P_j, we set *weight* to 1. Otherwise, we set *weight* to 0.5.

$$Csim(C_i, P_j) = \frac{2 \times level(MSCA(C_i, P_j))}{level(C_i) + level(P_j)} \times weight \qquad (1)$$

The similarity values between the target concept C_i and each P_j on the Kadokawa thesaurus hierarchy are shown in Figure 5. These similarity values are computed using Equation 1. For example, in '*CCI-type 0*' part calculation, the relation between the concept codes 274 and 26 corresponds to the relation between C_i and P_4 in Figure 5. So we assign the similarity 0.285 to the feature node labeled by 26. As another example, the concept codes 503 and 50 have a relation between C_i and P_2 and we obtain the similarity 0.857. If more than two concept codes exist in one CCI-type, such as <*type 0*: *078, 274*>, the maximum similarity value among them is assigned to the input node, as in Equation 2.

In Equation 2, C_i is the concept code of the feature set, and P_j is the concept codes in the <*CCI-type*: *concept codes*> pair which has the same CCI-type as C_i.

$$InputVal(C_i) = \max_{P_i}(Csim(C_i, P_j)) \qquad (2)$$

The use of concept similarity scheme gives another advantage. By adopting this concept similarity calculation, we can achieve a broad coverage of the method. If we use the exact matching scheme instead of concept similarity, we may obtain only a few concept codes matched with the features. Consequently, sense disambiguation would fail because of the absence of clues.

4.3 Learning of Disambiguation Models

Using the obtained feature set and training patterns, we learned 4 types of disambiguation models, such as neural network, decision tree, support vector machine and majority voting system. Neural network and decision tree have been used in a lot of pattern recognition problems because of their strong capability in classification. And recently, support vector machine have generated a great interest in the community of machine learning due to its excellent generalization performance in a wide variety of learning problems.

From a statistical point of view, if the size of sample is small, generating different classifiers about the sample and combining them may result in more accurate prediction of new patterns. On the other hand, based on a computational view, if the sample is large enough, the nature of learning algorithm could lead to getting stuck in local optima. Therefore, a classifier combination is a way to expand the hypothesis space to represent the true

Table 3. Evaluation Results for Decision Tree with Different Pruning Levels

Pruning Confidence Level	**Precision** (correct / applied)
Level = 10%	76.26% (546/716)
Level = 15%	77.38% (561/725)
Level = 25%	77.30% (555/718)
Level = 40%	76.72% (547/713)

(※ number of test samples : 942)

Table 4. Evaluation Results for Support Vector Machine with Different Kernel Functions

Kernel Function	**Precision** (correct / applied)
Linear	79.89% (556/696)
Polynomial (degree=2)	80.60% (565/701)
Polynomial (degree=3)	79.92% (569/712)
RBF (width=0.5)	80.80% (568/703)
RBF (width=1.0)	80.66% (563/698)
RBF (width=2.0)	79.71% (554/695)

(※ number of test samples : 942)

function (Ardeshir, 2002). In our experiment, we adopted a majority voting system for combining base classifiers. A majority voting selects the sense of the test pattern based on receiving more than half votes of base classifiers.

To find the best parameters for decision tree and support vector machine, we evaluated performance of each classifier on various parameters. For this evaluation, we used 942 test samples extracted from KIBS (Korean Information Base System) corpus. Table 3 and 4 are the evaluation results for decision tree and support vector machine respectively and we selected parameters which showed the best performance. The parameter settings for our system are listed below.

[Decision Tree (DT)]
- *C4.5* Decision Tree Generator
- Pruning confidence level : 15%

[Neural Network (NN)]
- 2-layer network

[Support Vector Machine (SVM)]
- *SVM light*
- Kernel : RBF (width = 0.5)

[Majority Voting (MV)]
- Base classifiers : DT, NN, SVM

5 Experimental Evaluation

Our WSD approach is a hybrid method, which combines the advantage of knowledge-based and corpus-based methods. Figure 6 shows our overall WSD algorithm. For a given homograph, sense disambiguation is performed as follows. First, we search a collocation dictionary. The Korean-to-Japanese translation system COBALT-K/J has an MWTU (Multi-Word Translation Units) dictionary, which contains idioms, compound words, collocations, etc. If a collocation of the target word exists in the MWTU dictionary, we simply determine the sense of the target word to the sense found in the dictionary. This method is based on the idea of 'one sense per collocation'. Next, we verify the selectional restrictions of verbs described in the dictionary. If we cannot find any matched patterns for selectional restrictions, we apply the machine learning classifiers. If we fail in all the previous stages, we assign the most frequently appearing sense in the training corpus to the target word.

For an experimental evaluation, 15 Korean noun homographs were selected, along with a total of 1,200 test sentences in which one homograph appears (2 senses : 12 words, 3 senses : 2 words, 4 senses : 1 word). The test sentences were randomly selected from the KIBS corpus.

The baseline results are shown in Table 5, where the result A is the case when the most frequent sense was taken as the answer and the result B is the case when COL and SR stages were applied previously. Symbols COL, SR, ML and MFS in Table 5 and 6 indicate 4 stages of our method in

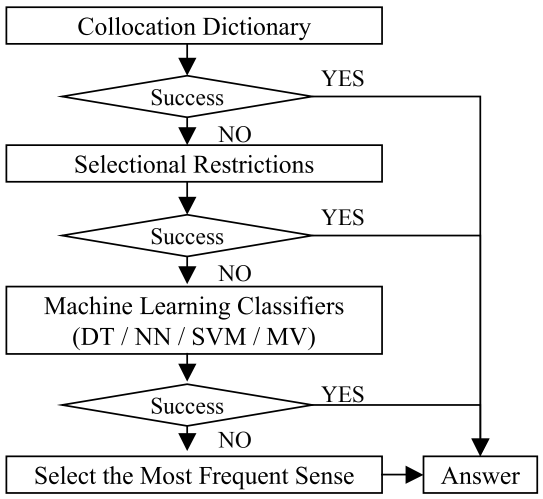


Figure 6. The Proposed WSD Algorithm

Table 5. Baseline Performance
※ Precision (correct # / applied #)

Stage	**Baseline A**	**Baseline B**
COL	N/A (0/0)	100% (21/21)
SR	N/A (0/0)	91.14% (216/237)
MFS	62.75% (753/1200)	64.23% (605/942)
Total	**62.75%** (753/1200)	**70.17%** (842/1200)

Table 6. Comparison Results of Classifiers
※ Precision (correct # / applied #)

Stage	**Model 1** [DT]	**Model 2** [NN]	**Model 3** [SVM]	**Model 4** [MV]
COL	100% (21/21)			
SR	91.14% (216/237)			
ML	77.38% (561/725)	78.93% (558/707)	80.80% (568/703)	81.84% (568/694)
MFS	53.00% (115/217)	57.87% (136/235)	51.46% (123/239)	51.61% (128/248)
Total	**76.08%** (913/1200)	**77.58%** (931/1200)	**77.33%** (928/1200)	**77.75%** (933/1200)

Table 7. Comparison Results of Classifiers for Each Word ※ Precision (correct # / applied #)

Word	**DT**	**NN**	**SVM**	**MV**
kasa	93.24% (69/74)	77.46% (55/71)	77.94% (53/68)	89.39% (59/66)
kancang	88.93% (47/56)	85.25% (52/61)	88.89% (48/54)	87.72% (50/57)
keli	79.17% (19/24)	57.69% (15/26)	76.92% (30/39)	83.33% (20/24)
kyengki	69.39% (34/49)	77.27% (34/44)	70.21% (33/47)	70.83% (34/48)
kyengpi	84.21% (32/38)	75.56% (34/45)	76.19% (32/42)	77.27% (34/44)
kwutu	86.44% (51/59)	90.57% (48/53)	87.27% (48/55)	87.72% (50/57)
nwun	91.84% (45/49)	93.48% (43/46)	93.62% (44/47)	91.67% (44/48)
tali	52.94% (27/51)	52.38% (22/42)	54.29% (19/35)	52.63% (20/38)
pwuca	82.61% (57/69)	86.67% (39/45)	87.10% (54/62)	85.94% (55/64)
swumyen	66.67% (22/33)	65.38% (34/52)	83.33% (30/36)	80.56% (29/36)
yongki	62.07% (36/58)	83.33% (35/42)	73.33% (33/45)	75.56% (34/45)
uysa	81.82% (9/11)	78.00% (39/50)	78.00% (39/50)	83.33% (35/42)
yenki (3 senses)	52.08% (25/48)	68.75% (22/32)	66.67% (20/30)	65.52% (19/29)
censin (3 senses)	93.55% (58/62)	93.22% (55/59)	98.08% (51/52)	96.49% (55/57)
cenlyek (4 senses)	68.18% (30/44)	79.49% (31/39)	79.49% (31/39)	76.92% (30/39)

Figure 6, respectively. Table 6 is the comparison results of 4 machine learning classifiers. To compare models with the same condition, we controlled the number of test samples which each model is applied to about 700.

As shown in the table, the majority voting system showed the best performance above all other single classifiers and exceeded the baseline A by 15.00%. Even if we exclude the help of the collocation information and selectional restrictions described in the dictionary, we achieved the improvement of 7.58% over the baseline B. This result is very promising for real world MT systems and indicates that word sense disambiguation can be improved by classifier combination. Among the single classifiers, SVM was better than DT and NN (see ML stage in Table 6). Interestingly, however, when followed by MFS stage, NN overtook the performance of SVM.

The results of classifiers for each word are shown in Table 7. A shadowed cell indicates the best classifier on the word. Although the majority voting recorded the best results on only 2 words, it showed good results on other words steadily. We can recognize that the best classifier is different for each word. Some words have a decision tree as the best classifier and some have a neural network. From this observation, we guess that each word may have disambiguation property of its own and require a different machine learning method according to its property. So if we can identify the disambiguation characteristics of words, we will be able to improve the system performance by applying a different classifier for each word.

6 Conclusion

To resolve sense ambiguities in Korean-to-Japanese MT, this paper has proposed a practical word sense disambiguation method using concept co-occurrence information. We showed that sense-tagged Korean corpus can be generated easily by using Japanese corpus and a machine translation system. In an experimental evaluation, the proposed WSD model using a majority voting achieved an average precision of 77.75% with an improvement over the baseline by 15.00%. This result indicates that word sense disambiguation can be improved by combining base classifiers and the concept co-occurrence information-based approach is very promising for real world MT systems.

We plan further research on feature selection. Compared with the surface form information of lexical words, the concept codes are somewhat diluted information as clues for WSD. Thus we will be able to improve the performance of system if we add other features to our disambiguation model, such as lexical words and part of speech of surrounding words. Also, we have a plan to develop a new similarity measure to find the more feasible similarity values for our system.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center(AITrc).

References

Ardeshir, G. (2002) Decision Tree Simplification for Classifier Ensembles. PHD Thesis, University of Surrey, U.K.

Ide, N. and Veronis, J. (1998) *Word Sense Disambiguation: The State of the Art.* Computational Linguistics, Vol.24, No.1, pp.1-40

Li, H. F., Heo, N. W., Moon, K. H., Lee, J. H. and Lee, G. B. (2000) *Lexical Transfer Ambiguity Resolution Using Automatically-Extracted Concept Co-occurrence Information.* International Journal of Computer Processing of Oriental Languages, Vol.13, No.1, pp.53-68

McRoy, S. (1992) *Using Multiple Knowledge Sources for Word Sense Discrimination*. Computational Linguistics, Vol.18, No.1, pp.1-30

Ng, H. T. and Zelle, J. (1997) *Corpus-Based Approaches to Semantic Interpretation in Natural Language Processing*. AI Magazine, Vol.18, No.4, pp.45-64

Ohno, S. and Hamanishi, M. (1981) *New Synonym Dictionary*. Kadokawa Shoten, Tokyo

Smadja, F. (1993) *Retrieving Collocations from Text: Xtract*. Computational Linguistics, Vol.19, No.1, pp.143-177

Yarowsky, D. (1992) *Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora*. In *Proceedings, COLING-92.* Nantes, pp. 454-460

Towards Semantic-Based Overlap Measures for Question Answering

Diego Mollá
Centre for Language Technology
Macquarie University
Sydney, NSW 2109
Tel. +61 2 9850 9531
Fax +61 2 9850 9551
diego@ics.mq.edu.au

Abstract

In this paper we present an evaluation of overlap-based measures of similarity for sentences in the same language. The measures include syntactic and semantic information, and to that end they incorporate grammatical relations and flat logical forms. A full parser is required to build the above information. Separate extrinsic evaluations within the context of question answering have been made with two different parsers to test the impact of the parser and the overlap measures.

1 Introduction

Text-based Question Answering (QA) is a hot research topic and the increasing availability of electronic text will ensure that research in this area will continue for long. Much of the current research on QA focuses on the development of methodologies for processing relatively large volumes of text. For example, the competition-based QA track of the Text REtrieval Conference (TREC) (Voorhees, 2001) uses more than 3Gb of source text. Competing systems often exploit the data redundancy existing in the source text. Some of them even use the Web to increase the data redundancy (Brill et al., 2001; Clarke et al., 2001, for example). These systems typically trade accuracy for speed and avoid the use of intensive natural language processing techniques.

Most of the current QA systems are based on an architecture like that of Figure 1 (Hirschman and Gaizauskas, 2001; Voorhees, 2001). In an off-line or indexing stage, an **indexing** module analyses the text documents and creates a set of document images that will be used by the subsequent QA modules. In an on-line stage, a **question analysis** module classifies the question and determines the type of the expected answer. The question analysis module would typically return a list of named-entity types that are compatible with the question (for example a *who* question typically indicates people or organisations). The module may also produce an image of the question. This image may be similar in format to the document images and can range from a simple bag of words (Cooper and Rüger, 2000, for example) to a fairly complex logical form (Harabagiu et al., 2001; Elworthy, 2000, for example). Once the question is analysed, a **document preselection** module identifies the documents that are most likely to contain the answer. This module typically uses information retrieval techniques that rely on bag-of-words approaches and statistical information (Voorhees, 2001). A **filtering** module examines the resulting documents and selects or rewards the named entities that are compatible with the question type. A **scoring** module then performs a more intensive analysis and ranks the preselected named entities (an possibly surrounding text) according to their likelihood to contain the answer. The scoring system relies on the output given by the question analysis module and possibly the images of the preselected documents that were created during the off-line stage. There may be feedback loops between the document preselection, filtering, and scoring modules to increase the likelihood of find-

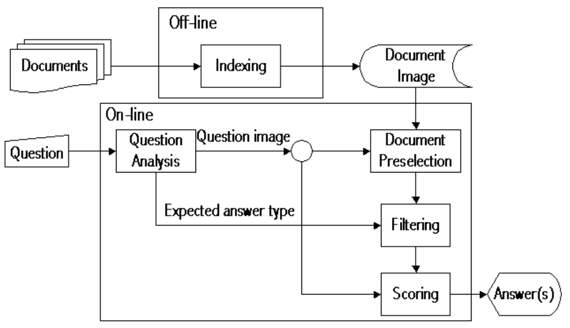


Figure 1: Architecture of a generic question-answering system.

ing difficult answers (Harabagiu et al., 2001, for example).

The scoring methodology used can be as simple as a word overlap or word frequency count, or as complex as an automatic proof system that operates on logical forms of both the questions and the answers. In some QA systems the scoring system relies heavily on the use of sentence patterns (Soubbotin, 2001, for example).

In the present study we have implemented a QA framework that uses simplifications of the modules described above. The emphasis in this study has been placed on the comparison of core methodologies for the scoring stage. To avoid the introduction of unwanted variables, we have avoided the use of methodologies that rely on world knowledge or domain knowledge. Thus, we have refrained from testing the use of external resources or inference systems.

With this QA framework we intend to assess the impact of syntactic and semantic information in a QA task. For that reason, we include information regarding word dependencies, grammatical relations, and logical forms in the procedure to measure the similarity between a question and an answer candidate. The results of the evaluations show both the impact of the scoring measures and the impact of the parsers used to extract the syntactic and semantic information. Section 2 describes grammatical relations and their use in an overlap measure. Section 3 focuses on the overlap of flat logical forms. Section 4 introduces the QA system that was used for the evaluations, and Section 5 explains the methodology used in the automatic evaluations. Finally, Section 6 discusses

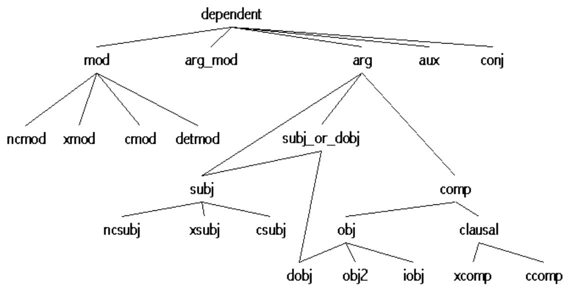


Figure 2: Hierarchy of grammatical relations.

the results and Section 7 concludes and points to future lines of research.

2 Grammatical Relations

We use the grammatical relations of Carroll et al. (1998), who devised them as a means to provide the canonical representation of the output of parsers for their evaluation. Figure 2 shows the hierarchical classification of the grammatical relations (Briscoe and Carroll, 2000). This hierarchy allows the mapping from the output of an arbitrary parser and therefore allow the evaluation of parsers with different output granularity.

Table 1 lists the grammatical relations used in this paper and the evaluation – For further detail about grammatical relations see (Briscoe and Carroll, 2000). For example, the grammatical relations for the sentence *The man that came ate bananas and apples with a fork without asking* are:

```
DETMOD(_,man,the),
CMOD(that,man,come),
SUBJ(come,man,_),
SUBJ(eat,man,_),
DOBJ(eat,banana,_),
DOBJ(eat,apple,_)
CONJ(and,banana,apple),
NCMOD(fork,eat,with),
DETMOD(_,fork,a),
XCOMP(without,eat,ask)
```

Briscoe and Carroll's grammatical relations are not to be confused with the dependency arcs used in the theory of dependency grammar (Mel'čuk, 1988). To illustrate the difference, consider the sentence *The man that came ate bananas and*

Relation	*Description*
CONJ(type,head+)	Conjunction
MOD(type,head,dependent)	Modifier
CMOD(type,head,dependent)	Clausal modifier
NCMOD(type,head,dependent)	Non-clausal modifier
DETMOD(type,head,dependent)	Determiner
SUBJ(head,dependent,initial_gr)	Subject
OBJ(head,dependent,initial_gr)	Object
DOBJ(head,dependent,initial_gr)	Direct object
XCOMP(head,dependent)	Clausal complement without an overt subject

Table 1: Grammatical relations used in this paper.

apples with a fork. Figure 3 shows the graphical representation of the structure returned by Conexor FDG, a dependency-based parsing system (Tapanainen and Järvinen, 1997). In dependency grammar a unique head is assigned to each word, thus the head of *man* is *ate*. However *man* is the dependent of more than one grammatical relation, namely `SUBJ(eat,man,_)` and `SUBJ(come,man,_)`. Furthermore, in dependency grammar a word can have at most one dependent of each argument type, and so *ate* can have at most one object. But the same is not true for grammatical relations, and we get both `OBJ(eat,banana,_)` and `OBJ(eat,apple,_)`. Thus, grammatical relations provide a sentence representation that is closer to the semantic contents of a sentence than the representation provided by dependency arcs.

Mollá and Hutchinson (2003) used the grammatical relations to compare the accuracy of two broad-coverage dependency-based parsers, Link Grammar (Sleator and Temperley, 1993) and Conexor Functional Dependency Grammar (Tapanainen and Järvinen, 1997) — henceforth referred to as Conexor FDG. The evaluation used a subset of the original relations: `SUBJ`, `OBJ`, `XCOMP`, and `MOD`. This subset was used because of limitations of the output of the parsers and the algorithms for the automatic construction of the grammatical relations. Thus, the reduced grammatical relations for the example *The man that came ate bananas and apples with a fork without asking* is:

```
MOD(that,man,come),
SUBJ(eat,man,_),
SUBJ(come,man,_),
OBJ(eat,banana,_),
OBJ(eat,apple,_),
MOD(fork,eat,with),
XCOMP(without,eat,ask)
```

		Link Grammar	*Conexor FDG*
Precision	SUBJ	50.3%	73.6%
	OBJ	48.5%	84.8%
	XCOMP	62.2%	76.2%
	MOD	57.2%	63.7%
	Average	***54.6%***	***74.6%***
Recall	SUBJ	39.1%	64.5%
	OBJ	50%	53.4%
	XCOMP	32.1%	64.7%
	MOD	53.7%	56.2%
	Average	***43.7%***	***59.7%***

Table 2: Intrinsic evaluations of Link Grammar and Conexor FDG.

The results of the evaluation on a corpus annotated with the correct grammatical relations (Carroll et al., 1998) show significantly higher values of recall and precision for Conexor FDG with respect to Link Grammar (Table 2).

The grammatical relations can be used by the scoring module of our QA system. We only need to compute the overlap of grammatical relations between the question and the answer candidate. In theory, we must use the hierarchical organisation of the grammatical relations to decide if two grammatical relations unify. For example, `SUBJ(eat,man,_)` should unify with

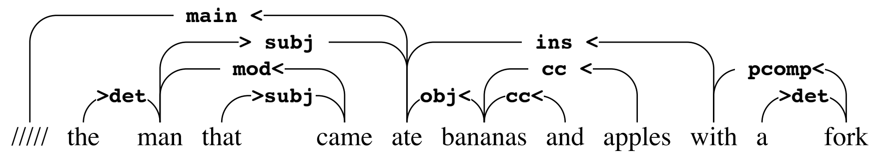


Figure 3: Dependency structure of a sample sentence.

SUB_OR_DOBJ(eat,man). However, since the same parser was used for both the question and the answer, the granularity of grammatical relations will be practically the same. Thus, each grammatical relation can be seen as an unstructured token and the scoring module can simply count the number of common tokens, very much like counting the overlap of words. This was the approach used in our QA prototype.

3 Flat Logical Forms

Flat logical forms have been used in several NLP systems including question-answering systems (Harabagiu et al., 2001; Lin, 2001; Mollá et al., 2000, for example). The flat logical forms that we use in our QA system are borrowed from (Mollá et al., 2000), who uses reification to flatten out nested expressions. For example, the flat logical form of *The cp command will quickly copy files* is:[1]

```
object('cp',o2,[x2]),
object('command',o3,[x3]),
compound_noun(x2,x3),
prop('quickly',p5,[e6]),
evt('copy',e6,[x3,x7]),
object('file',o7,[x7])
```

Flat logical forms express the main predicate-argument dependencies between the entities introduced by the sentence in a form that is suitable for computing semantic similarity. In particular, we only need to find the common predicates between the question and the answer candidate. The only additional complexity is the handling of variables in the terms of the question. It is therefore necessary to instantiate the question variables with constants found in the answer candidate. For example, the logical form of *Which command copies files?* is (the symbols in uppercase indicate variables):

```
object('command',O1,[X1]),
evt('copy',E2,[X1,X2]),
object('file',O2,[X2])
```

If this logical form is to match that of the sentence *The cp command will quickly copy files* above, the scoring module needs to instantiate the variable O1 in the question with the constant o3 in the answer candidate, X1 with x3, and so on. In our implementation we have used Prolog unification.

Since there are several plausible combinations of variable instantiations, the scoring module finds the set of instantiations that provides the highest overlap of logical forms.

Table 3 shows the flat logical forms of questions that differ solely in the argument positions, the flat logical form of an answer candidate, and the resulting overlaps.

4 The Question Answering Framework

In contrast with (Mollá et al., 2000), the semantic interpreters used in our evaluations to compute the logical forms do not use any additional lexical, domain, or world knowledge. Furthermore, there is no disambiguation step and there is no anaphora resolution module. The resulting semantic interpreters may therefore be less accurate, but the resulting QA systems are in a better position to be compared with the QA systems based on grammatical relations described in Section 2. Once it is decided which methodology is better, it is conceivable to add the additional modules that further enhance the expressivity of the sentence image.

For the present evaluation we used the Remedia Publications Reading Comprehension corpus used by DeepRead (Hirschman et al., 1999). The corpus is aimed at testing the degree of reading comprehension by children, and the documents

[1]For illustration purposes, the logical forms used in this paper are slightly different from the ones shown in the literature.

Answer candidate	**Flat Logical Form**
John saw Mary	object('john',o1,[x1]), object('mary',o3,[x3]), evt('see',e2,[x1,x3])

Question	**Flat Logical Form**
Did John see Mary?	**object('mary',O,[X]), evt('see',E,[Y,X]), object('john',O2,[Y])**
Did Mary see John?	object('john',O,[X]), **evt('see',E,[Y,X])**, object('mary',O2,[Y])

Table 3: Question answering using flat logical forms. Overlap shown in bold.

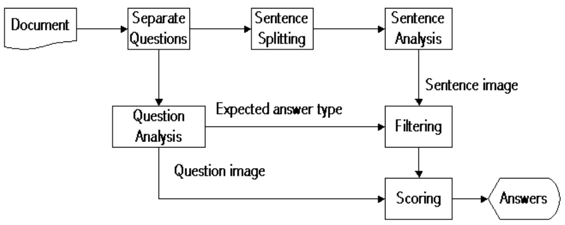


Figure 4: Architecture of the question-answering system.

Regex	*Expected Answer Type*
`^Who`	person, organization
`^What`	any
`^When`	date, time
`^Where`	location
`^Why`	any

Table 4: Question types and expected answer types.

in this corpus are classified into several levels of reading proficiency. There are about 30 documents for each of the levels 2, 3, 4, and 5, with a total of 117 documents. Each document includes a short piece of text and five questions (*who-*, *what-*, *when-*, *where-*, and *why-*) about the text.

The corpus contains annotations of the coreference chains and the named entities. The answers are also marked-up in the text and a gold standard for every answer is available. These annotations make the corpus suitable for the development and test of QA systems.

To evaluate the impact of the parsers and the scoring modules we have developed a simple question answering system framework (Figure 4). Given that every document contains the questions that are to be asked about the document, our QA system does not need to include a preselection stage. Instead, every document is processed independently. The system has a pre-processing stage that segments the document into text and questions. The text is split into sentences and each sentence is analysed independently.

Every question in the document is analysed to produce the question image. The question is also classified into one of the *who-*, *what-*, *when-*, *where-*, and *why-* categories. Depending on the question category, the question analysis module determines and returns the likely named entities of the expected answers.

Table 4 shows the named entities associated with each question type. The question classifier is extremely simple due to the fact that there are always five questions, and they have a very simple pattern. Thus, the regular expressions shown in Table 4 suffice to identify the question types. The procedure to determine the expected answer type is therefore very simple but fairly effective for the corpus and the named-entity types used in the named-entity annotations. Since the focus of this work is on the comparison of the scoring modules we did not feel we needed to produce a more sophisticated question analysis module.

The named-entity annotations provided with the evaluation corpus relieves the system from using a named-entity extraction component. The resulting system performs perfect named-entity extraction, which may artificially enhance the final quality of the answers returned. The results therefore cannot be compared with the results given by QA systems such as those participating in the Question Answering track of TREC (Voorhees, 2001), but they are good for the purposes of comparison that we pursue here.

The answer extraction module penalises all sentences that have no entities compatible with the answer type by giving them an initial score of -100. The only part of the QA system that varies across the comparisons is the scoring component.

The scoring methods described above can therefore be compared free of interference from other modules.

Following the specifications of the QA track of TREC8 to TREC10, the system returns the five answer candidates with highest scores, ranked by scores in descending order.

5 Evaluation Methodology

All four combinations of parser (Link Grammar and Conexor FDG) and scoring module (grammatical relations and flat logical forms) were used in the evaluations. All other modules in the QA system were left untouched.

The evaluation of the quality of the answers was done automatically. An answer candidate is judged correct if more than 80% of its words appear in one of the correct answers provided by the corpus annotations. We have not evaluated the accuracy of this evaluation method, but we have no reasons to believe that the final conclusions of the comparison are significantly affected by this automatic evaluation procedure, since all the experiments were evaluated with the same procedure.

The final measure is TREC QA's Mean Reciprocal Rank (MRR). Thus, for a given question, if the first correct answer returned is in rank □, the question is scored as □, or ! if no correct answer is found. The final score of the system is the mean of scores for the individual questions.

An initial inspection of the first results revealed that the overall scores were very low. As a result, several answer candidates would receive the same score. Table 5 shows an example of the answers returned for a question. We can observe that all five answer candidates were given the same score, but only the last candidate was correct. There is no way for the system to know which of the sentences with same score is best. To determine the impact of returning several sentences with the same score, two indices were computed in addition to the MRR. The two indices correspond to the MRR that would result if the correct answer was chosen first or last among those of the same score. These indices represent the "best" and "worst" case, respectively, and ideally they would be almost equal.

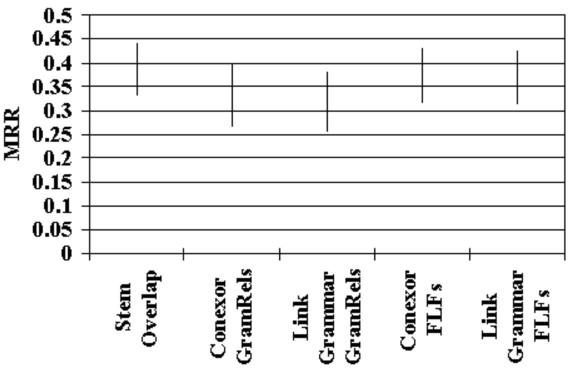


Figure 5: Evaluation of scoring measures.

6 Results and Discussion

The vertical bars in Figure 5 show the variation between the best and worst MRR for the text of levels 3, 4, and 5 together. Five cases are displayed, corresponding with a baseline scoring based on stem overlap and all combinations of parser (either Conexor FDG or Link Grammar) and scoring measure (either overlap of grammatical relations or overlap of flat logical forms).

Overall, we can see that the flat logical forms give better results than the grammatical relations. This is not surprising, since the flat logical forms were designed for tasks that require the semantic comparison of sentences. In contrast, the grammatical relations were designed for the comparison of parsers.

Also, Conexor FDG produces better results than Link Grammar. These results confirm Mollá and Hutchinson (2003)'s findings that Conexor FDG is slightly better than Link Grammar in a similar QA system that is based on a different corpus and evaluation methodology. Furthermore, the increase of performance with Conexor FDG is only marginal. This is in contrast with a direct comparison between Link Grammar and Conexor FDG which, as shown in Table 2 above, rated Conexor FDG much better than Link Grammar. The current experiment shows that, regardless of the method used (grammatical relations or flat logical forms), the MRR of the QA system that uses Conexor FDG is only slightly higher than the MRR of the QA system that uses Link Grammar. Thus, our results present further evidence that intrinsic evaluations

Rank	*Sentence*	*Score*	*Overlap*	*Correct*
1	1989 Remedia Publications, Comprehension s-4	1	`compound_noun(v_x2,v_x3)`	no
2	(North Redwood, Minn	1	`compound_noun(v_x2,v_x3)`	no
3	Not long ago, he ran an ad in some newspapers In small towns	1	`object(ad,v_o7,[v_x7])`	no
4	The ad showed a drawing of lovely furniture	1	`object(ad,v_o2,[v_x2])`	no
5	The ad said the furniture was for sale	1	`object(ad,v_o2,[v_x2])`	yes

Table 5: Answers returned for the question *What does the Sears ad offer?*

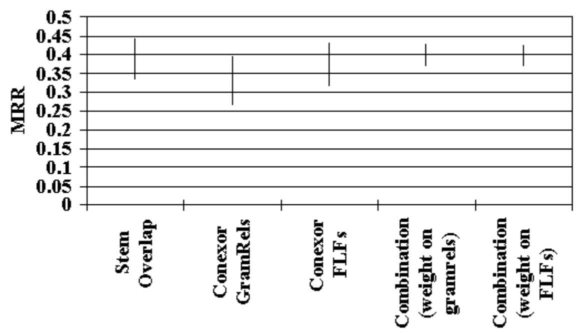


Figure 6: Combination of overlap measures.

are of very limited value, as stated already by Galliers and Sparck Jones (1993). An extrinsic evaluation that shows the impact of the modules to evaluate within the context of an application (QA in this case) may give results that differ substantially from those of an intrinsic evaluation.

What is surprising is the fact that, as Figure 5 shows, a measure based on simple stem overlap gives better results. Subsequent experiments indicate that a combination of the above scoring systems plus overlaps of word dependencies and word forms produce better results and they have a narrower difference between best and worst cases. For example, Figure 6 shows the results of two possible combinations:

Weight on grammatical relations:

□ □ "O

Weight on flat logical forms:

□ "O □

In the above formulas, stands for word form overlap, is the overlap of dependencies, is the overlap of minimal logical forms, and □ is the overlap of grammatical relations.

The dependencies used were extracted from the Conexor FDG parser, which is dependency-based. All the experiments about the combination of scoring systems were used with Conexor FDG. Similar results are expected with Link Grammar or other parsers. We are experimenting with the impact of other possible scoring combinations.

7 Conclusions and Further Research

This research shows that the combined information on word dependencies, grammatical relations and flat logical forms improves the accuracy of the system with respect to the individual measures, though the additional resources required to extract syntactic and semantic information may not justify the use of these measures against simple word overlap.

Further research is necessary to determine the reason for this. For example, it may be that the very nature of the Reading Comprehension document set makes it unlikely to represent real-world text. The text is intentionally simple, the documents are short and there is little text redundancy. Furthermore, the fact that the texts are of varied topics and that every document contains the questions that apply to the document makes it highly unlikely that the questions associated to a specific document have eligible answer candidates outside the document. For all these reasons we are performing similar experiments with the corpus used in the QA track of TREC 10 and TREC 11. However, preliminary results support the results presented in this paper.

It may well be that one needs to compute more complex overlap measures to leverage the additional information. Additional further research includes the evaluation of weighted overlap measures that consider the relative importance of specific grammatical relations or logical form terms, and the determination of the optimal weights to be given.

Finally, perhaps the simple questions used in the Reading Comprehension corpus and in TREC do not require the use of much linguistic information. An evaluation framework with more complex questions is necessary to test this possibility.

We also plan to evaluate the impact of other NLP modules such as anaphora resolvers and disambiguators, and the use of lexical resources (e.g. WordNet or localisations of WordNet) and domain and world knowledge. The current QA system can be easily expanded to allow for all of these evaluations.

References

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Ted Briscoe and John Carroll. 2000. Grammatical relation annotation. On-line document. `http://www.cogs.susx.ac.uk/lab/nlp/carroll/grdescription/index.html`.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC98*.

C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2001. Web reinforced question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Richard J. Cooper and Stefan M. Rüger. 2000. A simple question answering system. In Voorhees and Harman (Voorhees and Harman, 2000).

David Elworthy. 2000. Question answering using a large NLP system. In Voorhees and Harman (Voorhees and Harman, 2000).

Julia R. Galliers and Karen Sparck Jones. 1993. Evaluating natural language processing systems. Technical Report TR-291, Computer Laboratory, University of Cambridge.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's question-answering server. In Voorhees and Harman (Voorhees and Harman, 2001).

Lynette Hirschman and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Read: A reading comprehension system. In *Proc. ACL'99*. University of Maryland.

Jimmy J. Lin. 2001. Indexing and retrieving natural language using ternary expressions. Master's thesis, MIT.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Diego Mollá and Ben Hutchinson. 2003. Intrinsic versus extrinsic evaluations of parsing systems. In *Proc. European Association for Computational Linguistics (EACL), workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, Budapest, April. Association for Computational Linguistics, ACL.

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *Traitment Automatique des Langues*, 41(2):495–522.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

M. M. Soubbotin. 2001. Patterns of potential answer expression as clues to the right answers. In Voorhees and Harman (Voorhees and Harman, 2001).

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Procs. ANLP-97*. ACL.

Ellen M. Voorhees and Donna K. Harman, editors. 2000. *The Ninth Text REtrieval Conference (TREC-9)*, number 500-249 in NIST Special Publication. NIST.

Ellen M. Voorhees and Donna K. Harman, editors. 2001. *The Tenth Text REtrieval Conference (TREC-10)*, number 500-250 in NIST Special Publication. NIST.

Ellen M. Voorhees. 2001. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.

The Importance of High Quality Input for WSD: An Application-Oriented Comparison of Part-of-Speech Taggers

Tanja Gaustad
Humanities Computing
University of Groningen
P.O. Box 716
9700 AS Groningen, The Netherlands
Tel +31-(0)50-363 59 77
Fax +31-(0)50-363 68 55
`T.Gaustad@let.rug.nl`

Abstract

In this paper, we present an application-oriented evaluation of three Part-of-Speech (PoS) taggers in a word sense disambiguation (WSD) system. Following the intuition that high quality input is likely to influence the final results of a complex system, we test whether the more accurate taggers also produce better results when integrated into the WSD system. For this purpose, a stand-alone evaluation of the PoS taggers is used to assess which tagger is the most accurate. The results of the WSD task, computed on the training section of the Dutch Senseval-2 data, including the PoS information from all three taggers show that the most accurate PoS tags do indeed lead to the best results, thereby verifying our hypothesis. A surprising result, however, is the fact that the performance of the complex WSD system with the different PoS tags included does not necessarily reflect the stand-alone accuracy of the PoS taggers.

1 Introduction

Certain NLP tools are typically used as a subcomponent or a pre-processor in a more complex system, rather than as a complete application in their own right. A typical example of such tools are Part-of-Speech (PoS) taggers. What is usually not taken into account is the fact that the quality (in terms of accuracy) of each subpart of a complex system is likely to influence the final results considerably. Lately, standardized evaluation of NLP resources has gained more importance in the field of Computational Linguistics (e.g. CLEF workshops in information retrieval, Parseval, Senseval), but a tendency towards more application-oriented evaluation is only beginning.

In this paper, we will proceed to an application-oriented comparison of three PoS taggers in a word sense disambiguation (WSD) system. We will evaluate to what extent differences in stand-alone PoS accuracy influence the results obtained in the complex WSD system using the acquired PoS information. Since the Dutch data we use is not only ambiguous with regard to meaning but also with regard to PoS, accurate PoS information is very important to achieve high disambiguation accuracy.

The paper is structured as follows: We will start with a detailed description and comparison of the three PoS taggers including a stand-alone evaluation in order to compare their performance independently of the application to the WSD task. Then follows a description of the WSD system in which (the output of) the different PoS taggers will be incorporated and tested. This includes a presentation of the machine learning algorithm employed for classification (maximum entropy) and its application to WSD, as well as a note on the data and the settings used for the reported experiments. Next, the application-dependent results of the three PoS taggers will be presented and discussed. We end the paper with conclusions and some ideas for future work.

2 Comparison of Part-of-Speech Taggers

The PoS taggers we compare in this article are:

- a Hidden Markov Model tagger (section 2.1),
- a Memory-Based tagger (section 2.2),
- a transformation-based tagger (section 2.3).

We chose these three taggers because they were readily available, could easily be trained for Dutch without major changes in the architecture, and represent distinct, widely used types of existing PoS taggers.

All three taggers were trained on the Dutch Eindhoven corpus (uit den Boogaart, 1975) using the WOTAN tag set (Berghmans, 1994). The original WOTAN tag set, consisting of 233 tags, was too detailed for our purpose. Instead, we used the limited WOTAN tag set of 48 PoS tags developed by (Drenth, 1997) for training and testing in the stand-alone comparison.

In the context of our WSD application, however, we are only interested in the main PoS categories. Therefore, we discarded all additional information from the assigned PoS tags in the WSD corpus. This resulted in 12 different tags being kept: Adj (adjective), Adv (adverb), Art (article), Conj (conjunction), Int (interjection), Misc (miscellaneous), N (noun), Num (numeral), Prep (preposition), Pron (pronoun), Punc (punctuation), and V (verb).[1]

For the stand-alone results, 80% of the training data was actually used for training, 10% for tuning (setting of parameters, etc.) and the accuracy was computed on the remaining 10%. Note that the results of the stand-alone comparison solely serve to illustrate the difference in performance observed independently of an application in order to be able to assess the added value of a more accurate PoS tagger in the WSD application.

2.1 Hidden Markov Model PoS Tagger

The first PoS tagger we used is the trigram Hidden Markov Model (HMM) tagger (Prins and van Noord, 2003) developed in the context of 'Alpino', a natural language understanding system for Dutch (Bouma et al., 2001; van der Beek et al., 2002).[2]

[1]See table 2 for a distribution of the main PoS tag categories in the WSD data and the Eindhoven corpus.

[2]See `http://www.let.rug.nl/~vannoord/alp`.

In this standard trigram HMM, each state corresponds to the previous two PoS tags and the probabilities are directly estimated from the labeled training corpus (Manning and Schütze, 1999). There are two types of probabilities relevant in this model, the probability of a tag given the preceding two tags $P(t_i|t_{i-2}t_{i-1})$ as well as the probability of a word given its tag $P(w_i|t_i)$.

These probabilities are computed for each tag individually. Training the HMM with the forward-backward algorithm, we can calculate $P(t_i = t)$ for all potential tags:

$$P(t_i = t) = \alpha_i(t)\beta_i(t)$$

where $\alpha_i(t)$ is the total (summed) probability of all paths through the model that end at tag t at position i, and $\beta_i(t)$ is the total probability of all paths starting at tag t in position i continuing to the end. Comparing all the values for $P(t_i = t)$, unlikely tags are removed.

Smoothing of the trigram probabilities is achieved through a variant of linear interpolation (Collins, 1999) where lower order models are also taken into account and weights are assigned to each of the models to capture their relative importance.

Since the tagger's lexicon has been created from the training data, the test data very likely contains unknown words which means that no initial set of possible tags can be assigned to these words. Two different strategies have been incorporated in the HMM tagger used here. First, a heuristic rule for recognizing names has been added which assigns an N tag to all capitalized words.[3] Second, a set of automata (also created on the basis of the training data) is used to find possible tags based on the suffixes of unknown words (Daciuk, 2000).

2.2 Memory-Based PoS Tagger

The second tagger we have used in the experiments reported here is the Memory-Based Tagger (MBT) (Daelemans et al., 2002a).[4] It is a PoS tagger based on Memory-Based Learning, an extension of the k-Nearest-Neighbour approach, which has proved to

[3]Words in sentence initial position are decapitalized beforehand.

[4]Freely available for research purposes at `http://ilk.uvt.nl/software.html`.

be successful for a number of languages and NLP applications (Zavrel and Daelemans, 1999; Veenstra et al., 2000; Hoste et al., 2002).

MBT consists of two components: a memory-based learning component and a performance component for similarity-based classification. During classification, the similarity between a previously unseen test example and the examples in memory is computed using a similarity metric. The category of the test example is then extrapolated based on the most similar example(s).

Given an annotated corpus, three data structures are automatically extracted: a lexicon, a case base for known words, and a case base for unknown words. During tagging, each word is looked up in the lexicon and, if it is found, its lexical representation is retrieved and its context determined. The resulting pattern is disambiguated using extrapolation from the nearest neighbours in the known words case base. If a word is not present in the lexicon, its lexical representation is computed on the basis of its form, its context is determined, and the resulting pattern is disambiguated using extrapolation from nearest neighbours in the unknown words case base. In both cases, the output is a best guess of the category for the word in its current context.

For the known words, the preceding two tags and words as well as the ambiguous tag and word to the right of the current position have been used to construct the known words case base. Classification was achieved using the IGTREE algorithm with one nearest neighbour. For unknown words, the preceding tag, the ambiguous tag to the right, as well as the first and the last three letters of the ambiguous word itself were taken into account to construct the unknown words case base. For classification, the IB1 algorithm with 9 nearest neighbours was used. In both cases GainRatio feature weighting was applied. For details on the different possible algorithms see (Daelemans et al., 2002b).

2.3 Transformation-Based PoS Tagger

As the third member of the comparison, we used a Brill-style transformation-based tagger (TBL) (Brill, 1995) for Dutch (Drenth, 1997). The main components of a transformation-based tagger are a specification of admissible transformations and a learning algorithm. Interdependencies between words and tags are modeled by starting out with an imperfect tagging which is gradually transformed into one with fewer errors. This is achieved by selecting and sequencing transformation rules using the learning algorithm.

In an initial step, each word is assigned a tag independent of context. A known word is assigned its most likely tag determined by a maximum likelihood estimation from the training corpus. An unknown word, on the other hand, is assigned a tag based on lexical rules learned during training. All unknown words are initially tagged N. The application of lexical rules adapts the tag (where necessary) based on the local properties of the unknown word, such as its suffix.

After each word has received an initial tag, contextual rules are applied changing the initial PoS tag (where necessary) based on the context of the word to be tagged. The best contextual transformation rules and their order of application are selected by the learning algorithm during training.

The present implementation of the TBL PoS tagger for Dutch uses around 250 lexical rules and 300 contextual rules.

PoS Tagger	Accuracy
TBL	94.20
HMM	95.93
MBT	**96.21**

Table 1: Stand-alone results (in %) for the three PoS taggers on 10% of the Eindhoven corpus data

2.4 Stand-Alone Results for the PoS Taggers

As we have mentioned earlier, the stand-alone results for the PoS taggers were computed using 80% of the Eindhoven Corpus (containing a total of 760,000 words) for training and 10% for tuning. The accuracy shown in table 1 was computed on the remaining 10% of the corpus.

We can clearly see that the MBT tagger is performing best, followed by the HMM tagger, the least accurate tagger being the TBL tagger.[5]

If the hypothesis that more accurate input to complex systems will produce more accurate results is

[5]All results differ significantly applying the paired sign test with a confidence level of 95%.

correct, then these stand-alone results raise the expectation that when applying all three taggers in our WSD system—with all other settings being equal—accuracy should be highest when the MBT tagger was used to tag the data. Performance is expected to decrease with the use of the HMM tagger and to be lowest for the TBL tagger.

This expectation might be falsified by the (possible) corpus dependency of the three PoS taggers: the capacity to generalize from the training corpus to the corpus to be tagged might be bigger in one tagger than in another, which means that the results obtained in the complex system can diverge from the expectation raised by the stand-alone results.

Let us now turn to the application in which we will use the three PoS taggers presented and evaluated above.

3 Word Sense Disambiguation for Dutch

Semantic lexical ambiguity remains a major problem in natural language processing (NLP) for which to date no satisfactory solution has been found. Word sense disambiguation (WSD) refers to the resolution of lexical semantic ambiguity and its goal is to attribute the correct sense(s) to words in a certain context. Accurate disambiguation of word senses is important for e.g. machine translation, information retrieval or document extraction.

The WSD system used in these experiments is a supervised corpus-based algorithm combining statistical classification with different kinds of linguistic information. This system explores the intuition that (high quality) linguistic information is beneficial for WSD. PoS is definitely one of the more accessible sources of linguistic knowledge. The hypothesis behind comparing various PoS taggers in this application is that the quality of the PoS tags assigned to the data can significantly influence the accuracy obtained by our WSD system.

In contrast to the English WSD data, the Dutch Senseval-2 WSD data is ambiguous with regard to PoS. This means that accurate PoS information is even more important since the WSD system is supposed to do morpho-syntactic as well as semantic disambiguation.

We will now first explain the statistical classification algorithm used and then proceed to describe the WSD system, its settings as well as the corpus used to generate the comparative results.

3.1 Maximum Entropy Classification

The statistical classifier used in the experiments reported here is a *maximum entropy classifier* (Berger et al., 1996). Maximum entropy is a general technique for estimating probability distributions from data. If nothing about the data is known, it involves selecting the most uniform distribution where all events have equal probability. In other words, it means selecting the distribution which maximises the entropy.

If data is available, labeled training data is seen as a number of features which are used to derive a set of constraints for the model. This set of constraints characterises the class-specific expectations for the distribution. So, while the distribution should maximise the entropy, the model should also satisfy the constraints imposed by the labeled training data. A maximum entropy model is thus the model with maximum entropy of all models that satisfy the set of constraints derived from the training data.

The maximum entropy model is built using the following formula:

$$p(c|x) = \frac{1}{Z} exp\left(\sum_i \lambda_i f_i(x,c)\right)$$

where the property function $f_i(x,c)$ represents the number of times feature i is used to find class c for event x, and the weights λ_i are chosen to maximise the likelihood of the training data and, at the same time, maximise the entropy of p.

This means that during training the weight λ_i for each feature i present in the training data is computed and stored. During testing, the sum of the weights λ_i of all features i found in the test instances is computed for each class c and the class with the highest score is chosen.

The main advantage of maximum entropy modeling is that the property functions, including all the different types of (linguistic) information in the model, take into account any information which might be useful for disambiguation. Thus, dissimilar types of information can be combined into a single model for WSD and no independence assumptions (as in e.g. a Naive Bayes algorithm) are necessary.

3.2 Corpus and System Settings

The corpus used in this evaluation is the Dutch Senseval-2 corpus[6] (see (Hendrickx and van den Bosch, 2001) for a detailed description). In the experiments reported here, we only made use of the *training section* of the Dutch Senseval-2 dataset, containing approximately 120,000 tokens and 9,300 sentences.

In a first step, the corpus is lemmatized and PoS tagged. Then, for each ambiguous wordform/lemma[7] all instances of its occurrence are extracted from the corpus. These instances are then transformed into different feature vectors. So a feature vector of the ambiguous wordform 'aarde' (earth/soil) corresponding to the model which comprises all possible information (incl. PoS) and uses context words would look like this:

```
aarde N gat in de , zodat het aarde_grond
```

where the first slot represents the lemma, the second the PoS, the third to eighth slot are the context words (left before right) and the last slot represents the sense or class.[8] Only context words within the same sentence as the ambiguous wordform/lemma were taken into account. If for instance there was no left context, it was filled with "empty" features. Varying the information included, different feature sets are constructed.

For the *basic classifier* based on ambiguous wordforms, the feature set contains the corresponding lemma as well as a context of three words to the left and to the right of the ambiguous word. For the basic classifier based on ambiguous lemmas, the corresponding wordform and the context are included. The context can either be composed of wordforms or lemmas. For the *classifiers including PoS tags*, we in addition include the PoS tags of the ambiguous wordform/lemma from the various PoS taggers.

On the basis of the different feature sets, separate classifiers are built for every ambiguous wordform or lemma. This implies that the basis for grouping occurrences of particular ambiguous words together is that either their wordform or their lemma is the same. In the experiments presented here, a frequency threshold of 10 was used, which means that classifiers were only built for the wordforms with an amount of training instances equal to or above the threshold. For the remaining wordforms, the baseline count was used, thus assigning the most frequent sense to every instance.

In total, there were 1,364 ambiguous lemmas in the corpus of which 622 presented 10 or more occurrences, and 952 ambiguous wordforms of which 486 had 10 or more occurrences. So 622 lemma classifiers and 486 wordform classifiers were built.

The context was treated as a 'bag of words' which means that the position of a context word relative to the ambiguous wordform was not taken into account. This approach was chosen to help limit the data sparseness problem: if the context features are all treated dependent on their position relative to the ambiguous word in the sentence, the model will have more features to assign weights to. This means that the sparse data problem will be worse. If, on the other hand, context features are "lumped" together independent of their relative position, there are less features to be estimated and there is more data for the particular feature 'context'.

4 Results and Evaluation of the WSD Application

Before we turn to the actual results of using the different PoS taggers in our WSD system for Dutch, let us first compare the differences regarding the assigned PoS tags. Table 2 shows the distribution of the different PoS tags in the WSD data depending on the PoS tagger used, as well as the distribution of the PoS tags in the training corpus.

A major difference between the distribution of PoS tags is that both the HMM and MBT tagger assign more V tags, whereas the TBL tagger assigns more N tags. The preference for N tags in the TBL tagger can be explained by the fact that all unknown words initially get tagged N. Also, in Dutch verbal infinitives have the same morphological suffix as plural nouns (*-en*). INT and Misc differ with all three taggers, but we could not detect any obvious reason for this. As we can see from table 2, there

[6]For more information on Senseval and for downloads of the data see `http://www.senseval.org/`.

[7]A wordform/lemma is 'ambiguous' if it has two or more different senses in the training data. The sense '=' is seen as marking the basic sense of a word/lemma and is therefore also taken into account.

[8]'Sense' or 'class' refers to the different labels which disambiguate the ambiguous wordforms/lemmas.

PoS	TBL		HMM		MBT		Train. Corpus
N	22,830	(19.46%)	20,041	(17.08%)	20,384	(17.37%)	20.35%
Punc	19,792	(16.87%)	20,151	(17.17%)	20,142	(17.17%)	12.69%
V	17,645	(15.04%)	19,505	(16.62%)	19,556	(16.66%)	15.13%
Pron	13,880	(11.83%)	13,938	(11.88%)	13,885	(11.83%)	9.82%
Adv	11,250	(9.58%)	11,289	(9.62%)	11,178	(9.53%)	8.19%
Art	9,477	(8.08%)	9,350	(7.96%)	9,328	(7.95%)	9.39%
Prep	8,190	(6.98%)	8,358	(7.26%)	8,229	(7.01%)	10.54%
Conj	6,713	(5.72%)	6,742	(5.74%)	6,770	(5.77%)	5.18%
Adj	6,313	(5.38%)	6,621	(5.63%)	6,626	(5.65%)	6.53%
Num	869	(0.74%)	713	(0.61%)	744	(0.63%)	1.78%
Int	376	(0.32%)	559	(0.47%)	455	(0.39%)	0.18%
Misc	3	(0.003%)	71	(0.04%)	41	(0.04%)	0.22%

Table 2: Frequencies of PoS tags assigned by each PoS tagger in the WSD data and distribution of PoS in the training corpus

are bigger differences between the TBL tagger and the other two, whereas the differences between the HMM and the MBT tagger are less noticeable.

In order to test the real error of the classifiers built, we used a leave-one-out approach (Weiss and Kulikowski, 1991; Manning and Schütze, 1999). This means that every data item in turn is selected once as a test item and the classifier is trained on all remaining items. The accuracy of a single classifier is then the number of data items correctly predicted. The overall accuracy is the total of data items correctly predicted by all classifiers.

The results in table 3 show the average accuracy on our training data using leave-one-out as a test method with respectively wordforms and lemmas as basis.

As the table of results shows, the WSD system performs well. The basic classifiers containing a minimum of information already do significantly better than the frequency baseline.[9] Furthermore, adding PoS as extra linguistic information—next to the lemma/wordform and the context already included in the basic classifiers—does increase results over the accuracy achieved with a basic classifier. This supports the underlying hypothesis behind the WSD system that more linguistic information is beneficial for WSD. Since the WSD data needs to be disambiguated morpho-syntactically as well as with regard to lexical semantic ambiguity, it is not surprising that adding PoS information achieves better results than only using the lemma/wordform and context.

Comparing the performance among the different PoS taggers, we can see quite clearly that our expectations are (partly) confirmed: the MBT tagger, which did best in the stand-alone evaluation, is also working best in the WSD system. This is the case for all setups: using wordforms or lemmas as basis for the classifiers, as well as for classifiers including context as wordforms or as lemmas.[10]

Surprisingly enough, the hypothesis does not hold for the "ranking" of the HMM and TBL taggers. Despite the fact that the HMM tagger performed second best in the stand-alone evaluation, it does not perform better than the TBL tagger when integrated into the WSD system.

A possible explanation might be that the difference between the training corpus and the WSD data is so big that the HMM tagger is no longer more accurate than the TBL tagger in the WSD application, leading to the conclusion that the HMM tagger is more corpus dependent than the TBL tagger. A possible reason might be that the heuristics for unknown

[9]Assigning the most frequent sense to every occurrence of an ambiguous wordform/lemma.

[10]Applying the paired sign test with a confidence level of 95%, all results using MBT PoS tags were found to be statistically significantly better than results with other PoS tags (and than the basic classifiers). The classifiers including TBL and HMM PoS tags do not differ significantly from each other, but both perform significantly better than the basic classifiers.

Base: Wordforms			
Feature set	Accuracy		
baseline	76.70		
lemma, con. words (basic)	80.81		
lemma, con. lemmas (basic)	80.52		
	TBL	HMM	MBT
lemma, pos, con. words	81.67	81.67	**81.89**
lemma, pos, con. lemmas	81.42	81.36	**81.67**

Base: Lemmas			
Feature set	Accuracy		
baseline	73.41		
word, con. words (basic)	82.52		
word, con. lemmas (basic)	82.25		
	TBL	HMM	MBT
word, pos, con. words	83.32	83.34	**83.46**
word, pos, con. lemmas	83.06	83.05	**83.30**

Table 3: WSD results (in %) comparing the effect of integrating the output of different POS-taggers into a complex system

words in the HMM tagger produces worse results on the WSD data than the heuristics used by the TBL tagger. Since no gold-standard PoS tagged version of the WSD data exists, it is difficult to investigate this puzzle any further.

Nevertheless, our hypothesis that highly accurate input influences the results of a complex system is at least partly verified: the most accurate PoS tags also produce the most accurate results when integrated into our WSD system.

5 Conclusion and Future Work

In this paper, we tested the hypothesis whether high quality input improves the final results of a complex NLP system. We have therefore proceeded to an application-oriented evaluation of three PoS taggers in a WSD system. A transformation-based tagger, a Hidden Markov Model tagger, and a memory-based tagger were compared for this purpose.

After the MBT tagger has been established as the most accurate tagger in a stand-alone evaluation, the PoS information from all three taggers is integrated into our WSD system for Dutch. This supervised system uses maximum entropy classifiers which allow to integrate various sources of information into a single model.

The results computed on the training part of the Dutch Senseval-2 corpus show that the MBT tagger also produces the best results in the WSD system. This clearly indicates that highly accurate input into a WSD system is producing better results than qualitatively lesser input.

A surprising result, however, was the fact that the performance of the complex WSD system with the different PoS tags included does not necessarily reflect the stand-alone accuracy of the PoS taggers. Even though the HMM tagger performed better than the TBL tagger in the stand-alone comparison, there is no significant difference to be observed in the results of the WSD system. A possible explanation might be corpus dependency.

For future work, we would like to include the PoS tags of the context wordforms or lemmas to see whether our hypothesis still holds then. It would also be interesting to see whether the overall results are further improved by this additional information.

Acknowledgments

This research was carried out within the framework of the PIONIER Project *Algorithms for Linguistic Processing*. This PIONIER Project is funded by NWO (Dutch Organization for Scientific Research) and the University of Groningen. We are grateful to Robbert Prins for his help with the HMM tagger as well as to Gertjan van Noord and Menno van Zaanen for comments and discussions.

References

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Johan Berghmans. 1994. WOTAN—een automatische grammaticale tagger voor het Nederlands. Master's thesis, Nijmegen University, Nijmegen.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra, and Jakub Zavrel, editors, *Computational Linguistics in the Netherlands 2000*, pages 45–59, Amsterdam. Rodopi.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Computer and Information Science Department, University of Pennsylvania, Philadelphia.

Jan Daciuk. 2000. Finite state tools for natural language processing. In *Proceedings of the COLING 2000 Workshop "Using Toolsets and Architectures to Build NLP Systems"*, pages 34–37, Centre Universitaire, Luxembourg.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2002a. MBT: Memory-Based tagger, reference guide. Technical Report ILK 02-09, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University, Tilburg. version 1.0.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2002b. TiMBL: Tilburg Memory-Based learner, reference guide. Technical Report ILK 02-10, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University, Tilburg. version 4.3.

Erwin W. Drenth. 1997. Using a hybrid approach towards Dutch part-of-speech tagging. Master's thesis, Humanities Computing, University of Groningen, Groningen.

Iris Hendrickx and Antal van den Bosch. 2001. Dutch word sense disambiguation: Data and preliminary results. In *Proceedings of Senseval-2, Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 13–16, Toulouse.

Véronique Hoste, Walter Daelemans, Iris Hendrickx, and Antal van den Bosch. 2002. Evaluating the results of a Memory-Based word-expert approach to unrestricted word sense disambiguation. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pages 95–101, Philadelphia.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge.

Robbert Prins and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement automatique des langues*. forthcoming.

Pieter uit den Boogaart. 1975. *Woordfrequenties in Geschreven and Gesproken Nederlands*. Oosthoek, Scheltema en Holkema, Utrecht.

Leonoor van der Beek, Gosse Bouma, Rob Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In Mariët Theune, Anton Nijholt, and Hendri Hondorp, editors, *Computational Linguistics in the Netherlands 2001*, pages 8–22, Amsterdam. Rodopi.

Jorn Veenstra, Antal van den Bosch, Sabine Buchholz, Walter Daelemans, and Jakub Zavrel. 2000. Memory-Based word sense disambiguation. *Computers and the humanities*, 34(1-2):171–177.

Sholom Weiss and Casimir Kulikowski. 1991. *Computer Systems that Learn*. Morgan Kaufman, San Mateo.

Jakub Zavrel and Walter Daelemans. 1999. Recent advances in Memory-Based part-of-speech tagging. In *VI Simposio Internacional de Communicación Social*, pages 590–597, Santiago de Cuba.

Conceptual Schema Approach to Natural Language Database Access

In-Su Kang, Seung-Hoon Na, Jong-Hyeok Lee
Div. of Electrical and Computer Engineering
Pohang University of Science and Technology (POSTECH)
Advanced Information Technology Research Center (AlTrc)
San 31, Hyoja-dong, Nam-gu, Pohang, 790-784, R. of KOREA
{dbaisk, nsh, jhlee}@postech.ac.kr
fax: +82-54-279-5699

Abstract

Natural language database interfaces require translation knowledge to convert user questions into formal database queries. Previously, translation knowledge acquisition heavily depends on human specialties such as NLP, DBMS and domain engineering, consequently undermining domain portability. This paper attempts to semi-automatically construct translation knowledge by introducing a physically-derived conceptual database schema, and by simplifying translation knowledge into two structures – class-referring documents and class-constraining selection restrictions. Based on these two structures, this paper proposes a noun translation method that employs an information retrieval framework.

1 Introduction

A natural language database interface (NLDBI) allows users to access database data in a natural language (Androutsopoulos et al., 1995). In a typical NLDBI system, a natural language question is analyzed into an internal representation using linguistic knowledge that normally includes *domain knowledge* to reduce analysis ambiguities. The internal representation is then translated into a target database query by applying *mapping information* (Androutsopoulos et al. 1995) that associates the analysis results with target database structures. This paper uses *translation knowledge* to refer to both domain knowledge and mapping information, because both are required to translate a natural language question into a database query.

Previous approaches can be classified according to the extent that translation knowledge is integrated into linguistic knowledge. Tightly-coupled approaches hard-wire translation knowledge into linguistic knowledge in the form of semantic grammars (Hendrix et al., 1978; Waltz 1978; Templeton and Burger 1983). This approach shows a good performance for a particular domain. However, in adapting to other domains, new semantic grammars should be created with a considerable effort (Allen 1995).

In order to improve domain portability, many researchers have concentrated on isolating translation knowledge from linguistic knowledge through loosely-coupled approaches. These approaches can be further classified according to the extent that question analysis is performed. Syntax-oriented systems (Ballard et al., 1984; Damerau 1985; Lee and Park 2002) analyze questions up to a syntactic level, after which translation knowledge is applied to generate a database query. Logical form systems (Warren and Pereira 1982; Grosz et al., 1987; Alshawi et al., 1992; Androutsopoulos 1993; Klein et al., 1998) interpret a user question into a domain-independent literal meaning level.

Thus, in loosely-coupled approaches, transporting to a new database domain does not need to change linguistic knowledge at all, only tailoring translation knowledge to new domains. Even in this case, however, translation knowledge is difficult to describe. For example, syntax-oriented systems have to devise conversion rules that transform parse trees into database query expressions (Androutsopoulos et al. 1995), and logical form systems should define database relations for logical predicates. In addition, creating these translation knowledge demands considerable human expertise, such as NLP/DBMS/domain specialties. To reduce these manual interventions, many systems employ domain tools (Grosz et al. 1987) to collect domain vocabulary about target database structures, through a sequence of interactive procedures. However, these acquisition processes are passive, and time-consuming. Moreover, such tools cannot be easily adapted to other systems because they are customized to their own systems.

In order to automate translation knowledge acquisition for a new database, this paper attempts to semi-automatically construct translation knowledge by introducing a physically-derived conceptual database schema and by simplifying translation knowledge into two structures – a set of class/value documents having linguistic terms corresponding to domain classes, and a set of selection restrictions on domain classes. Based on these two structures, this paper proposes a noun translation method that employs an information retrieval framework.

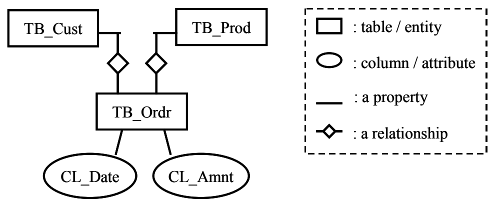
The remainder of this paper is as follows. The next two sections describe terminologies and a conceptual schema used in this paper. Section 4 explains an overview of a conceptual schema approach. Section 5 defines our translation knowledge. Section 6 details a noun translation strategy using translation knowledge, and concluding remarks are given in section 7. For representing Korean expressions, the Yale Romanization is used.

2 Terminologies

In this paper, a database object is defined as either a domain class or a domain class instance. A domain class refers to a table or a column in a database. A domain class instance indicates an individual column value. For example, suppose that a physical database contains two tables, TB_Customer and TB_Employee, and TB_Customer has a column C_cName, and TB_Employee has a column C_eCountry. All these are called domain classes. If the columns C_cName and C_eCountry have 'Abraham Lincoln' and 'France', respectively, as its values, each of these values is called a domain class instance.

A class term is defined as a lexical term referring to a domain class. A value term signifies a term indicating a domain class instance. For instance, the word 'customer' in a user question is a class term corresponding to the above domain class 'TB_Customer'. The word 'Lincoln' is a value term referring to the above domain class instance 'Abraham Lincoln'. In summary, a class term or a value term is used to indicate a word in a user question, and a domain class or a domain class instance is used to refer to a database object, such as a table, a column, or a column value.

3 Physical Entity-Relationship (pER) Schema



pER graph

Domain class	*Linguistic name*	*Definition*
TB_Cust	customer	Persons or companies that order products
TB_Prod	product	NULL
TB_Ordr	order	order information
CL_Date	date of order	dates when customers ordered products
CL_Amnt	amount of order	amount of products that customers ordered

Relationship	*Relationship description*
TB_Cust ⇔ TB_Ordr ⇔ TB_Prod	customers order products
	customers request orders

pER descriptions

Figure 1. Physical ER (pER) Schema

For a database, a conceptual schema like entity-relationship model structurally resembles a semantic network for the database domain. In addition, its components like entities, attributes, and relationships contain linguistic descriptions, which may bridge between natural language constructions and physical database structures. This paper tries to extract translation knowledge from a conceptual schema. However, a conceptual database schema is not always available. So, we define a physical Entity-Relationship (pER) schema as an approximation of real conceptual schema.

A pER schema is composed of a pER graph and its linguistic descriptions. A pER graph is structurally equivalent to physical database structures, where a node corresponds to a table or a column, and an arc defines a relationship between two tables, or a property between a table and its column. So a node in a pER graph is a domain class. Each node or arc contains linguistic descriptions that are called pER descriptions. As shown in figure 1, there are three kinds of pER descriptions - a linguistic name, definition, and relationship description.

A pER schema is created as follows. First, a logical schema is extracted from a target physical database. This reverse engineering process is automatically performed within a commercially available database modeling tool. The logical schema has the same structure as a physical database. So the logical schema becomes a pER graph. Next, domain experts provide linguistic descriptions for each component of the logical schema according to the following guidelines.

A linguistic name – in a noun phrase
A definition – in a definitional sentence
A relationship description – in a typical sentence including typical domain verbs

The input process is also graphically supported by the database modeling tool, and the pER descriptions can be automatically extracted by the modeling tool.

The term physical ER (pER) schema is used because it is an approximation of the target database's original ER (entity-relationship) schema in the sense that its structures are directly derived from a physical database. A pER graph is later used to generate conceptual query graphs. The pER descriptions have the potential to bridge between linguistic constructions and physical database structures. From pER descriptions, two translation knowledge structures are automatically generated, which will be described in section 5.

4 Conceptual Schema Approach

4.1 Domain Adaptation

Figure 2 shows an NLDBI architecture based on a conceptual schema. There are two processes; domain adaptation and question answering. For a new database domain, domain adaptation semi-automatically constructs translation knowledge. Translation knowledge is divided into two structures: a set of class/value documents corresponding to domain classes, and a set of selection restrictions on domain classes. A class document contains a set of class terms related to a domain class. Class terms are extracted from natural language descriptions of a pER schema.

A value document is created from a set of domain class instances associated with a column. Selection restrictions are also derived from linguistic descriptions of a pER schema.

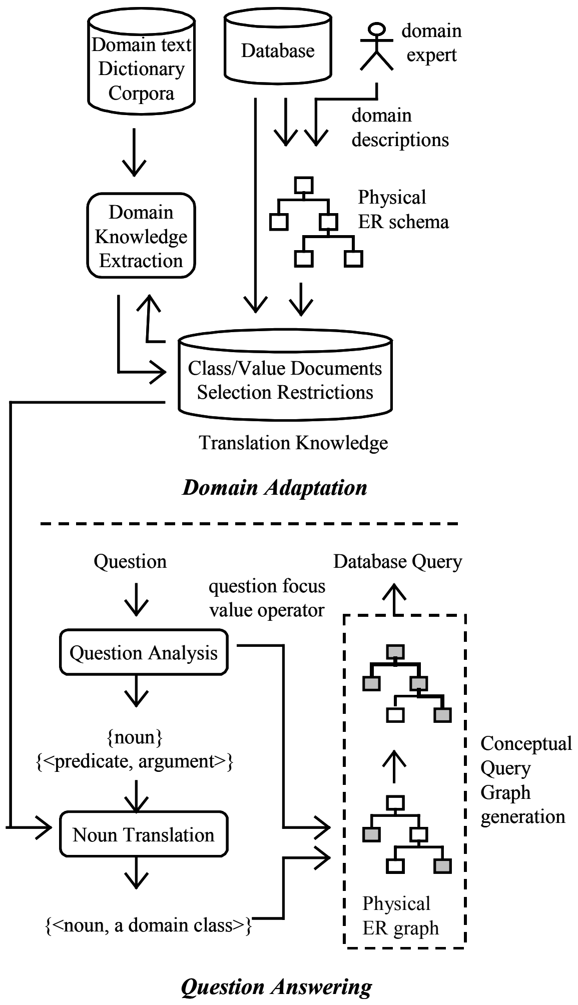


Figure 2. NLDBI architecture

The domain adaptation process may further augment this initial translation knowledge from other resources, such as domain materials, dictionaries, or corpora.

4.2 Question Answering

The question answering proceeds as follows. A user writes his or her information need in a natural language. A user question is then analyzed to produce a set of question nouns and a set of predicate-argument pairs. In Korean, these are obtained after morphological analysis, tagging, chunking, and partial dependency parsing. Question analysis also yields a set of feature-value pairs for each question noun. Among others, essential question features are a question focus and a value operator. These two features determine select-clause items and where-clause operators, respectively, in a final SQL query.

In noun translation, each question noun is considered as an IR query to retrieve lexically or semantically equivalent domain classes in the form of class or value documents. Afterwards, to each question noun holding two or more relevant domain classes, selection restrictions are applied to determine one correct domain class. Noun translation is explained in section 6.

Conceptual query graph generation creates a conceptual query graph (CQG) on the pER graph. First, domain classes produced by noun translation are marked on the pER graph, and question feature-value pairs are attached to the nodes of associated domain classes. On the pER graph, a CQG is searched, which is a connected subgraph connecting all the nodes that are attached with certain feature-value pairs.

The CQG is assumed to represent a domain-dependent question meaning, because nodes of the graph correspond to domain-dependent objects of question nouns, and arcs between nodes represent domain-dependent semantic relations between question nouns. Unlike the logical form method, this approach does not produce an intermediate domain-independent question meaning. Instead, a domain-dependent question meaning is directly represented in the form of a subgraph on a conceptual schema. So a final database query is generated from the graph. Given a CQG, database query generation is trivial. Entity nodes of the CQG go to SQL-from clause, and arcs between entities constitute SQL join operators. An SQL-select clause is obtained from question-focus-attached nodes. All value-operator-attached nodes are combined to create SQL-where conditions.

5 Translation Knowledge

5.1 Translation Knowledge Structures

Any NLDBI system demands translation knowledge, which consists of domain knowledge and mapping information. The former provides an analysis module with ambiguity-reducing devices, such as domain terminologies, domain-dependent selection restrictions, and a domain world model. The latter defines mappings between linguistic analysis results and target database structures. This paper divides translation knowledge into two structures. One is class-referring information, which is a collection of terms that directly refer to each domain class or domain class instance. The other is class-constraining information: a collection of selection restrictions on domain classes. Compared to the previous NLDBI translation knowledge, the first encodes both mapping information and domain terminologies, and the second corresponds to domain-dependent selection restriction. In addition, a pER graph plays the role of a domain world model.

5.2 Class-Referring Translation Knowledge

Formally, class-referring translation knowledge is defined as a set of pairs of *C* and *D*. *C* is a domain class, and *D* is a document that contains terms linguistically referring to *C*. *D* has two types; a class document and a value document. For each domain class, a class document is created from pER descriptions. In addition, for each domain class corresponding to columns, a value document is created from column data for the domain class. These documents are indexed to create a document collection to be used by the later noun translation module.

Class Document

A class document contains a set of lexically synonymous class terms for a domain class. Anticipated class terms are extracted from both linguistic names and definitions in a pER schema. A linguistic name *X* for a domain class is a noun phrase. In Korean, it is a compound noun optionally having a Korean genitive case marker '*uy*' (of). Its general form is $(N^+(uy)?\_)^*N^+$ in a regular expression, where _ is a word boundary and *N* is a simple noun.

A linguistic definition for a domain class is a definitional sentence, so it takes one of the following restricted forms in English translations.

(a kind of) + Y + adjective phrase modifying Y
X|it + be|mean|indicate|... + (a kind of) + Y + adjective phrase modifying Y

Both *X* and *Y* are class term candidates, since a taxonomic relation exists between them. *Y*, which may be also a compound noun, can be easily identified using a few patterns.

Class term extraction proceeds as follows. Given a compound noun, its genitive case markers are deleted, and each of the remaining compound nouns is segmented into a sequence of simple nouns. For example, $N_3N_2uy\_N_1$ is converted into $N_3+N_2+N_1$, where the last noun N_1 is a head of $N_3N_2uy\_N_1$ in Korean, and *uy* is a genitive case marker. Since different combinations of the simple nouns may constitute different question words to refer to the same domain class, a set of head-preserving compound nouns are generated from the simple nouns as follows.

N_3N_2 의$_N_1 \rightarrow N_3N_2+N_1 \rightarrow N_3+N_2+N_1 \rightarrow \{N_3N_2N_1, N_2N_1, N_1\}$

Since a head is an underlying concept of the compound noun, a head noun is preserved for all combinations.

Value Document

For each value term in a user question, an NLDBI system should determine the domain class to which it belongs. This value recognition problem (Templeton and Burger 1983) is critical since, unlike class terms, most value terms are open-ended. In addition, question value terms may take different forms from domain class instances stored in a database. For example, to refer to a database value *sam-seng-cen-ca* (Samsung Electronics) in Korean, users prefer partial forms like *sam-seng* (Samsung) in *sam-seng-ey-se cwu-mwun-han* (… that Samsung ordered).

To support partial matching between question value terms and domain class instances, this paper proposes n-gram value indexing. For each column of a target database, n-gram value indexing generates n-grams of the column data to create a value document. Among column data, linguistic terms are distinguished from alphanumeric terms.

For a linguistic term of k syllables, all-length n-grams from bi-grams to k-grams are generated as index terms of a value document in order to prepare all substrings expected as question value terms. For example, a column value *se-wul-thuk-pyel-si* is processed to generate these n-grams, *se-wul*, *wul-thuk*, *thuk-pyel*, *pyel-si*, *se-wul-thuk*, *wul-thuk-pyel*, *thuk-pyel-si*, *se-wul-thuk-pyel*, *wul-thuk-pyel-si*, *se-wul-thuk-pyel-si*, among which legitimate words as question terms are *se-wul*, *thuk-pyel-si*, *se-wul-thuk-pyel-si*.

On the other hand, generating n-grams for alphanumeric terms causes a severe storage problem. Damerau's method (Damerau 1985) reduces an open-ended set of alphanumeric terms into a closed set of patterns. Thus, it is adopted and slightly modified to include 2-byte characters like Korean. In the modified version, a canonical pattern *P* is defined as follows.

<P> ::= <U>{<U>}
<U> ::= [<C_1*>|<*C_2*>|<N>|<S>][1|2|...|255]*
where *<*C_k*> is a sequence of k-byte characters,*
<N> is a sequence of numbers,
<S> is a sequence of special characters.

For example, an alphanumeric database value *se-wul*-28@A-*ma* is converted into a canonical pattern, $C_22N2S1C_11C_21$. Next, in order to provide partial matching between patterns, a canonical pattern is decomposed into bi-grams. That is, for $C_22N2S1C_11C_21$, bi-grams $\_C_22$, C_22N2, N2S1, $S1C_11$, C_11C_21, $C_21\_$ are created and stored as index terms in a value document.

Pattern-based n-grams provide considerable storage reduction over storing canonical patterns, since canonical patterns are sliced into smaller n-grams that will have many duplicate n-grams. Hopefully, these n-grams provide partial matching capability even to the arbitrary alphanumeric terms.

5.3 Class-Constraining Translation Knowledge

$$K_v = \{\langle v, C_v\rangle\}, K_{cm} = \{\langle cm, C_{cm}\rangle\}$$

As class-constraining translation knowledge, two types of selection restrictions are defined for domain classes. K_v is a set of selection restrictions between domain verbs and domain classes. K_{cm} is a set of selection restrictions between surface case markers and domain classes. v is a verb appearing in pER descriptions, and C_v is a set of domain classes corresponding to arguments that v governs. cm is a surface case marker appearing in pER descriptions, and C_{cm} is a set of domain classes corresponding to arguments that cm attaches.

K_v and K_{cm} are extracted from predicate-argument pairs that are acquired by parsing pER descriptions. First, each predicate-argument pair is expanded to a triple of <*verb, noun, case marker*>. The *case marker* means a surface case marker of the *noun*. In Korean, the triple <*verb, noun, case marker*> is easily constructed from a predicate-argument pair, since each nominal argument has a surface case marker as a postposition within a word boundary. The second term of a triple is replaced by a *domain class* related to the noun. The modified triple is further divided into <*verb, domain class*> and <*case marker, domain class*>. Next, by merging a set of <*verb, domain class*> having the same *verb*, K_v is produced. Similarly, K_{cm} is obtained by merging a set of <*case marker, domain class*> having the same *case marker*. K_{cm} will be useful for value terms that correspond to different domain classes according to its case marker.

6 Noun Translation

After question analysis, a user question is analyzed into a set of question nouns and a set of predicate-argument pairs. Noun translation utilizes an IR framework to translate each question noun into a probable domain class. First, class retrieval converts each question noun into an IR query and retrieves relevant documents. Here, retrieved documents refer to candidate domain classes for the question noun, because each document is associated with a domain class. Next, class disambiguation selects a likely domain class among the candidate domain classes retrieved by class retrieval using predicate-argument pairs of the user question.

6.1 Class Retrieval

A question noun may be a class term or a value term, and a value term may be a linguistic value term or an alphanumeric value term. To be used as an IR query, these terms are converted into different vector queries, as these terms are differently treated in indexing class or value documents. That is, class terms are converted into word-based terms, linguistic value terms into a list of all-length n-grams, and alphanumeric value terms into a list of pattern-based n-grams. We employ three types of query representations; a conceptual vector for a class term, an all-length n-gram vector for a linguistic value term, and a pattern-based n-gram vector for an alphanumeric value term.

It is straightforward to distinguish whether a question noun is an alphanumeric term. However, it is nontrivial to distinguish between a class term and a linguistic value term, because many domain-dependent class terms are out-of-vocabulary words. So, for a question noun other than an alphanumeric term, class retrieval creates both a conceptual vector and an all-length n-grams vector, and retrieves documents for each query, and merges the retrieved documents. In the following, a conceptual vector representation for class terms is described.

If we simply convert a class term into a single term vector, it may cause a severe word mismatch problem (Furnas el al., 1987). Thus, the question noun is generalized to concept codes, which are then included in a vector query. Unfortunately, this method may risk obtaining mistaken similarity values if the correct concepts of the two terms are not similar while incorrect concepts of the two terms are similar. However, considering that domain terminologies show marginal sense ambiguities (Copeck et al., 1997), this concern will not be critical.

A query-document similarity is computed as follows.

$$Similarity(Q, D) = argmax_{t}\ W_Q(t) * W_D(t)$$

It simply selects the maximum value among weights of each matching term t. The reason is that, because all query terms belong to one homogeneous term group that originates from one lexical query term, the similarity between a query and a document means the best of similarities between the homogeneous group of a query term and homogeneous groups of several document terms.

6.2 Class Disambiguation

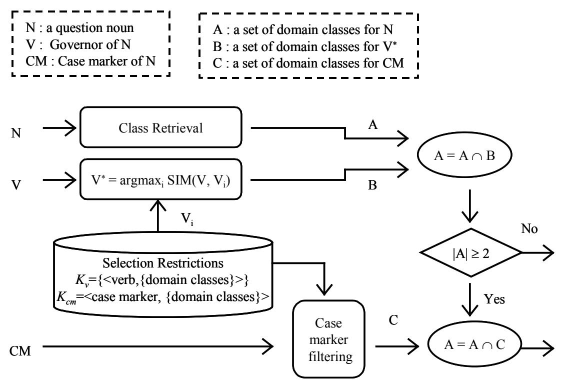

N : a question noun
V : Governor of N
CM : Case marker of N
A : a set of domain classes for N
B : a set of domain classes for V*
C : a set of domain classes for CM
N
Class Retrieval
A
A = A ∩ B
V
V* = argmax_i SIM(V, V_i)
B
V_i
No
|A| ≥ 2
Selection Restrictions
K_v={<verb,{domain classes}>}
K_cm=<case marker, {domain classes}>
Yes
CM
Case marker filtering
C
A = A ∩ C

Figure 3. Class Disambiguation

When question nouns are translated into domain classes, two types of ambiguities occur. Class term ambiguity occurs when a class term in a question refers to two or more domain classes. This ambiguity mostly results from general attributes that several domain entities share. For example, a question noun 'address' can refer to any 'address' attribute that the two entities 'customer' and 'employee' have at the same time. A value term ambiguity occurs when more than two domain classes share at least one domain class instance. Hence, date or numeric expressions almost always cause value term ambiguity. In particular, in an air flight domain, country names or city names will be shared by many domain classes, such as the location of departure and the location of arrival.

Class retrieval reduces the translation equivalents of each question noun to lexically or semantically equivalent domain classes. However, the above two ambiguities still remain after class retrieval. Class disambiguation resolves these ambiguities using class-constraining translation knowledge of K_v and K_{cm}. Disambiguation procedures proceed in two stages, as shown in figure 3.

In the first stage, for each question noun with two or more domain classes after class retrieval, K_v is searched to find a domain verb that is the most similar to the head verb of the question noun. The SIM value between two lexical words is the maximum of concept similarity values between all possible concept pairs of the two lexical words. Let B represent the set of domain classes associated with the domain verb, and let A be the set of domain classes retrieved by class retrieval for the question noun. Then, A is replaced by A intersection B. The effect is to reduce ambiguities by removing from A inconsistent domain classes that is not expected by a governor of the question noun.

The second stage takes the remaining ambiguities after applying K_v. K_{cm} is searched to find the same surface case marker as that of the question noun, and let C be the set of domain classes associated with the case marker. Then, A is further replaced by A intersection C. The effect is to select from A only the domain classes to which the case marker can attach.

For example, consider the following question.

Q1: sam-seng-ey-se *cwu-mwun-han cey-phwum-un ?*
E1: Show me products (cey-phwum) that Samsung(sam-seng) ordered (cwu-mwun-han) ?

A word *sam-seng-ey-se* consists of a root *sam-seng* and a postpositional case marker *ey-se*. Suppose that the question noun *sam-seng* retrieves three ambiguous domain classes {TB_Supplier, TB_Customer, TB_Shipper} by class retrieval. Then, using a governor *cwu-mwun-ha* of *sam-seng*, K_v is searched to find <*cwu-mwun-ha*, {TB_Customer, TB_Product, TB_Order.Amount, TB_Order.Date} >, and reduce ambiguity as follows.

A = {TB_Supplier, TB_Customer, TB_Shipper}
B = {TB_Customer, TB_Product, TB_Order.Amount, TB_Order.Date}
A = A ∩ B = {TB_Customer}

In this case, K_{cm} is not used. As another example, consider this question.

Q2: se-wul-ey-se len-ten-kka-ci pi-hayng-si-kan-un ?
E2: Show me the flight duration (pi-hayng-si-kan) from(ey-se) Seoul(se-wul) to(kka-ci) London(len-ten) ?

By class retrieval, a question noun *se-wul* will retrieve two domain classes {TB_Flight.Departure, TB_Flight.Arrival}. Unlike Q1, B will be empty since Q2 does not provide any verb. Then, using a case marker *ey-se*, K_{cm} is searched to find < *ey-se*, {TB_City, TB_Country, TB_Flight.Departure}>, and reduce ambiguity as follows.

A = {TB_Flight.Departure, TB_Flight.Arrival}
C = {TB_City, TB_Country, TB_Flight.Departure }
A = A ∩ C = {TB_Flight.Departure}

7 Conclusion

To effectively deal with the domain portability problem, this paper proposed the conceptual schema approach, which depends on the following three main components that differ from previous approaches.

The first is an introduction of a physical ER schema, which is easily created from a target database itself by domain experts with the help of a database modeling tool. The schema is used for capturing domain-dependent question meaning, because semantic constraints among domain objects are represented in the graph part of the ER schema. The second is the automatic construction of translation knowledge from a physical ER schema. To accomplish this, we defined two types of translation knowledge structures: a set of class-referring documents and a set of class constraining selection restrictions. The construction process requires only a shallow analysis of linguistic descriptions for a physical ER schema. The third is a noun translation strategy based on an information retrieval framework, where question nouns are associated with domain classes, lexically or semantically.

In future, we will extend the current translation knowledge from other resources, such as domain materials, dictionaries, and corpora.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center(AITrc).

References

Allen J. 1995. *Natural Language Understanding*. Redwood City, CA:Benjamin Cummings.

Alshawi, H., Carter, D., Crouch, R., Pulman, S., Rayner, M., and Smith, A. 1992. CLARE – A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. *Final report*, SRI International.

Androutsopoulos, I. 1993. Interfacing a Natural Language Front-End to Relational Database. *Master's thesis,* Technical Report 11, Department of Artificial Intelligence, University of Edinburgh.

Androutsopoulos, I., Ritchie, G.D., and Thanisch, P. 1995. Natural Language Interfaces to Databases – An Introduction. *Natural Language Engineering,* 1(1):29-81.

Ballard, B.W., Lusth, J.C., and Tinkham, N.L. 1984. LDC-1: A Transportable, Knowledge-Based Natural Language Processor for Office Environments. *ACM Transactions on Office Information Systems* 2(1):1-25.

Copeck, T., Barker, K., Delisle, S., Szpakowicz, S., and Delannoy, J.F. 1997. What is Technical Text?. *Language Sciences* 19(4):391-424.

Damerau, F. 1985. Problems and Some Solutions in Customization of Natural Language Database Front Ends. *ACM Transactions on Office Information Systems* 3(2):165-184.

Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. 1987. The vocabulary problem in human-system communication. *Communications of the ACM* 30(11):964-971.

Grosz, B.J., Appelt, D.E., Martin, P.A., and Pereira, F.C.N. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence* 32(2):173-243.

Hendrix, G.G., Sacerdoti, D., Sagalowicz, D., and Slocum, J. 1978. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2):105-147.

Klein, A., Matiasek, J., and Trost, H. 1998.. The treatment of noun phrase queries in a natural language database access system. *Proceedings of the COLING-ACL'98 workshop on the computational treatment of nominals*, Montreal, Quebec, pp.39-45.

Lee, H.D., and Park, J.C. 2002. Interpretation of Natural language Queries for Relational Database Access with Combinatory Categorial Grammar. *International Journal of Computer Processing of Oriental Languages* 15(3):281-304.

Templeton, M., and Burger, J. 1983. Problems in Natural Language Interface to DBMS with Examples with EUFID. *Proceeding of the 1st Conference on Applied Natural Language Processing,* Santa Monica, California, pp.3-16.

Waltz, D.L. 1978. An English Language Question Answering System for a Large Relational Database. *Communications of the ACM* 21(7):526-539.

Warren, D., and Pereira, F. 1982. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *Computational Linguistics* 8(3-4):110-122.

Document Classification in Structured Military Messages

Oliver Carr and Dominique Estival
Human Systems Integration Group
Command and Control Division
Defence Science and Technology Organisation
{Oliver.Carr,Dominique.Estival}@dsto.defence.gov.au

Abstract

We present new results for the DSTO project on document classification of military messages. We report more specifically on the improvements to the Part-Of-Speech (POS) tagging, a probabilistic process that assigns a tag to a token, and discuss the training for Date Time Groups POS tags. A new implementation of the rule-based classifier is described. The results obtained on two databases of real military messages are encouraging and the document classification module has now been integrated with a query user interface.

1 Introduction

In (Carr and Estival, 2002), we presented the first tentative results of the Document Classification project we have been conducting at DSTO and we discussed the shortcomings of the approach we were using. In this paper, we present the results we have obtained in the continuation of that project, after having implemented improvements in the POS tagging component and taken a different approach for the rule-based classifier component. These results show that rule-based classifiers can give reasonable results for structured textual information, when using appropriate language models for POS tagging.

1.1 Goals of the project

A large part of the Defence Information Environment (DIE) used at the Deployable Joint Force Headquarters (DJFHQ) is based on Lotus collaborative and messaging applications. The staff members of DJFHQ use Lotus databases to log operational events and Lotus e-mail for actions and administrative functions. Around 200 messages per day are entered into these Lotus Notes log databases. Many DJFHQ staff members have expressed difficulty in finding particular information in their information reservoirs and our goal is to develop a more effective query interface between DJFHQ staff and their information reservoirs. This work already resulted in the development of the Query Building Interface (QBI), which was designed to create a better search interface to multiple log databases and to the users e-mail database. The rule-based Document Classifier we describe here has been trained and evaluated on Lotus operational log databases (OPS logs) from DJFHQ. It can now provide a categorisation for each document from the OPS logs and is integrated with QBI, as described in Section 7.2.

1.2 Proposed Architecture

Fig.1 below shows how QBI and the Document Classifier could be integrated in the existing IT infrastructure. In this new Server Environment, both QBI and the Document Classifier interface with the Lotus Notes database.

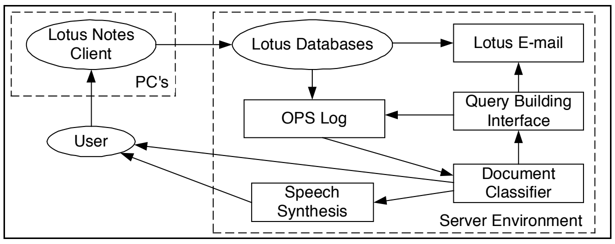


Fig.1 Proposed Architecture

In this architecture, users enter and access documents through Lotus Notes as they do now, and they receive notification of the document classification. One possible scenario is to use a Text-to-Speech module to warn of the arrival of some pre-specified document types, eg. NOTICAS (Notification of Casualty) or MEDSITREP (Medical Situation Report). For a NOTICAS, the injured person's details could be automatically retrieved and read out to the Commander or sent to a different display.

1.3 Structure of the system

The Document Classifier module in Fig.1 is named SOP-MRC (Standard Operating Procedures Rule Based Multiclass Classifier), and as shown in Fig.2, it consists of two main components: a Part-Of-Speech (POS) Tagger and a Classifier.

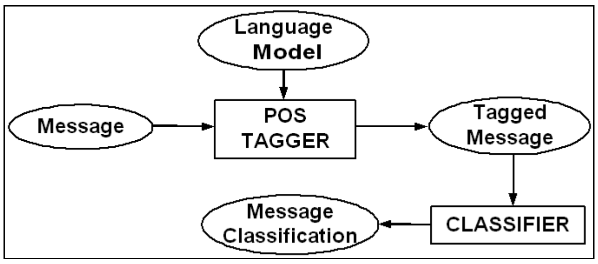


Fig. 2 The SOP-MRC module

The POS Tagger component, described in Section 3, is a probabilistic process that assigns a tag to a token. We also describe the training of this component in Section 3 and present our extension of the POS tagset for the military message data.

The Classifier component, described in Section 4, takes as input the list of pairs <token> <tag> produced by the POS Tagger for an incoming message and uses rules to determine the document type (including "free text") of the message.

We present in Section 5 the results we obtained on data from two military exercises. One database (VP-02) contains messages used to train the POS Tagger and to develop the classifier rules. The second database (TT-01) contains similar documents from another military exercise.

2 Shortcomings of the previous approach

There were two types of problems with the first approach we took to classify the SOP documents from the DJFHQ message database. The first one was that the data did not conform to expectations and the second one was that the classification rules were too brittle. Both issues have been addressed by the new approach to writing the classification rules described in Section 4.2.

DJFHQ operators use formatted text in the free text fields of their Lotus Notes operational log databases. This formatted text is defined by Standard Operating Procedure (SOP) documents, and there are 88 different SOP Document Types corresponding to different message types. Our first approach had been to define rules based on the definition of the SOPs, which are available to the operators writing those messages as MS Word documents. However, the actual messages often do not follow the format prescribed by the SOPs and, in addition, they often contain attachments and other material, which makes classification more difficult. The new rules now take into account variations in the way operators actually write their messages and allow more flexibility in the classification. This is described in Section 4.

Another problem was that the POS tagset used by our POS Tagger did not cover some token types that are very important in military messages. In particular, one lesson from our earlier work was that it is necessary to recognise Date Time Group (DTG) expressions and that we would have to develop our own tagset to fit the military domain. The additions we made to the POS tagset, are discussed in more detail in Sections 3.2 and 3.5.

3 POS Tagging

3.1 QTAG

The POS tagger we chose to use is Qtag, a portable trainable language-independent probabilistic tagger developed by the University of Birmingham (Mason, 2003; Tufis and Mason, 1998). There are several training corpora available on the Internet to train POS taggers.[1] Qtag was originally trained with the Industrial Parsing of Software Manuals (IPSM) (Sutcliffe et al, 1996), which uses the Penn Treebank tagset, and it comes with the Birmingham – Lancaster Tagset and the associated resource file trained for English.

Qtag takes free text as input and outputs SGML, with each line containing the tag and the token it corresponds to. An example of input from our corpus and of the output produced by Qtag is given in Fig. 3.

[1] See e.g. the Automatic Mapping Among Lexico-Grammatical Annotation Models (AMALGAM) project: http://www.comp.leeds.ac.uk/amalgam/amalgam/multi-parsed.html

Input (VP-02)	Qtag Output
From HQCLSC, HSS facilities allocated to CLSC as follows, A. 34 Fd Hosp (U.K)	<w pos="IN">From</w> <w pos="JJ">HQCLSC</w> <w pos=",">,</w> <w pos="NN">HSS</w> <w pos="NNS">facilities</w> <w pos="VBN">allocated</w> <w pos="TO">to</w> <w pos="NN">CLSC</w> <w pos="CS">as</w> <w pos="VBZ">follows</w> <w pos=",">,</w> <w pos="NN">A.</w> <w pos="CD">34</w> <w pos="NN">Fd</w> <w pos="NN">Hosp</w> <w pos="NN">(U.K)</w>

Fig. 3 Output of Qtag using original tagset

To deal with the particular type of text contained in SOP documents, 59 new POS tags (mainly formatting tags) were added to the original tagset of 45 tags. Fig. 4 shows the same text as Fig. 3, tagged by Qtag using the language model containing these additional domain specific tags.[2,3]

Input (VP-02)	Qtag Output
From HQCLSC, HSS facilities allocated to CLSC as follows, A. 34 Fd Hosp (U.K)	<w pos="From">From</w> <w pos="VB">HQCLSC</w> <w pos=",">,</w> <w pos="NN">HSS</w> <w pos="NNS">facilities</w> <w pos="VBN">allocated</w> <w pos="TO">to</w> <w pos="NN">CLSC</w> <w pos="as">as</w> <w pos="NPS">follows</w> <w pos=",">,</w> <w pos="FrmA">A.</w> <w pos="CD">34</w> <w pos="NN">Fd</w> <w pos="NN">Hosp</w> <w pos="JJ">(U.K)</w>

Fig. 4 Output of Qtag using new tagset

3.2 Date Time Groups (DTGs)

As was described in (Carr and Estival, 2002), the analysis of our previous results showed that they were unsatisfactory in part because the POS Tagger did not recognise Date Time Groups (DTGs), which are very common in our texts, and which play an important role in document type recognition for SOPs. For the purpose of document classification, a DTG is a single unit of information, but there are 3 types of DTGs that appear in the SOPs:

- DTG_S, with time and time zone information,
- DTG_M, with day, time and time zone information,
- DTG_L, with day, time, time zone, month, and year information.

Examples of these are given in (1) with their corresponding POS tag.

(1)	DTG_S	1259Z
	DTG_M	310745Z
	DTG_L	200830ZAUG02

Although the 3 different types of DTGs are not often distinguished by the classifier rules, the POS Tagger needs to be trained on these 3 different DTG types, to avoid confusion with other alphanumeric strings. (2) is an example of the output from Qtag, where the DTG_S tag is correctly assigned to the text "1100K".

```
(2)     <w pos="at">AT</w>
        <w pos="DTG_S">1100K</w>
        <w pos="NN">C130</w>
```

As shown in Table 1, the baseline performance of Qtag (trained on 80% of the VP-02 data and tested on the remaining 20%) for DTGs was fairly low. This is due to the inadequate training data for DTGs in this corpus, which comes from one military exercise covering a short period of time, and thus conatining few variations for dates.

	POS Baseline			
	DTG_S	DTG_M	DTG_L	All Tags
Recall	9.68%	15.72%	24.28%	74.39%
Precision	6.90%	13.74%	14.86%	75.23%

Table 1 Baseline Performance of Qtag

3.3 Training with additional data.

To improve recognition of DTGs, we decided to create additional examples of DTGs to boost the training data for Qtag. For each of the DTG types, additional data was created in a systematic way to obtain instances of DTGs covering a wider range of dates and times. Table 2 shows the performance of Qtag with additional training data for DTGs.

[2] HQCLSC stands for "Headquarters Combined Logistics Support Command". Note that the POS tags assigned to it (JJ in Fig.3 and VB in Fig.4) are incorrect, as are several of the other tags.

[3] The tag <FrmA>, meaning a "formatted A character", covers the strings "\nA.", "\n(A)" and "\nA)".

	additional DTG_M			
	DTG_S	DTG_M	DTG_L	All Tags
Recall	100.00%	98.91%	95.65%	90.81%
Precision	80.52%	98.91%	95.65%	85.94%
	additional DTG_S			
	DTG_S	DTG_M	DTG_L	All Tags
Recall	100.00%	98.91%	95.65%	92.75%
Precision	80.52%	98.91%	95.65%	87.77%
	additional DTG_L			
	DTG_S	DTG_M	DTG_L	All Tags
Recall	100.00%	98.91%	95.65%	90.95%
Precision	80.52%	98.91%	59.46%	85.94%

Table 2 Additional training data for DTGs

Since the worst performing category had been DTG_M, we first added 482 additional instances of DTG_M to the training file for Qtag. The same process was repeated for DTG_S and DTG_L, with 158 and 8,063 additional instances respectively.

3.4 Overtraining

Table 2 shows that the performance of Qtag improved when the DTG_M data was first added but decreased significantly after DTG_L data was added (additional DTG_S training data did not make any significant difference). The decrease in performance after the DTG_L data was added is due to overtraining of Qtag. Using the same recursive algorithm, adding year information leads to the creation of many more instances of DTG_L than DTG_M and DTG_S and skews the training data, resulting in many false positives for that category. Since the training text with the added DTG_M and DTG_S gave the best performance, this is what we used to create the Qtag language model.

3.5 New POS tags for measure units

Table 3 shows examples of DTG tokens that were miscategorised by Qtag. We can see that most of these are in fact genuine DTGs, which is good news since the classifier rules are not concerned with the type of DTG (DTG_S, DTG_M, or DTG_L) but only with the occurrence of a DTG.

Number	Example	Tag	Correct Tag(s)
1	030/02OF170015ZMAY02	NN	DTG_L
1	2100S	DTG_S	LAT_LONG_S
2	4000FT,	DTG_S	DST
8	4000L	DTG_S	WGT
5	171659Z	NN	DTG_M
13	WEST	req	NN
8	5000M	DTG_S	DST
32	(2)AT	NN	Frm2 at
7	(0.5-0.7	NN	CD
1	A.151206KMAR02	NN	FrmA DTG_L
2	PD:130800K	NN	Pd DTG_M
1	C.LAND	NN	FrmC VB

Table 3 Errors in POS tagging for DTGs

Further analysis of the miscategorisations shown in Table 3 suggests ways in which the performance of the POS Tagger can be improved:

§ add additional training data for DTGs with different minute information than 0 or 5;

§ add POS tags for measure units, such as <WGT> for weights, <DST> for distances, <SPD> for speeds,

§ add POS tags for the various different types of Latitude and Longitude information or Grid reference.

Some examples are given in (3).

(3) <w pos= "WGT">2500KG</w>
<w pos ="DST">500NM</w>
<w pos = LATLONG>15.35S/151.20E </w>

In the end, 71 extra tags were added to the tagset, giving a total of 116 POS tags.[2] The new Qtag language model was trained on 80% of the POS tags from the VP-02 data. The remaining 20% (36862 tags from 430 messages from VP-02) were used to test the performance of the POS Tagger. Table 4 shows the results obtained for the DTG_S, DTG_M and DTG_L tags, after Qtag was trained with the additional training data for the new measure units POS tags.

New Tags	Recall	Precision
DTG_S	100.00%	91.18%
DTG_M	98.91%	99.27%
DTG_L	97.34%	100.00%

Table 4 DTGs with new language model

[2] There were 57 tags for formatting, 3 for DTGs, 3 for measure units and 6 for Lat/Long/Grid. Only 111 out of the 116 different POS tags appear in our test data.

Table 5 shows the overall results for Qtag using the macroaverage and microaverage statistics as described in (Sebastiani, 2001). Almost half of the POS tags in the test data were <NN>. We believe using the microaverage result without <NN> gives a better indication of performance.

Recall and Precision Averages	**Recall**	**Precision**
Macroaverage	89.69%	95.65%
Microaverage	97.84%	97.08%
Microaverage (no NN)	92.67%	94.75%

Table 5 Overall Performance of Qtag

4 Rule-based Classifier

Unlike most work on document classification (see Jackson and Moulinier, 2002, or Manning and Schütze, 1999), we do not rely on the semantic content of the documents to classify our documents, but take advantage of the very highly constrained structure of the documents. This is an example of *Category-Pivoted Text Classification* where the classifier is given a classification and must find which messages should be assigned to a given class, as opposed to *Document-Pivoted Text Classification,* which tries to determine the appropriate classifications for a set of documents (Sebastiani, 2002).

Quoting from (Jackson and Moulinier, 2002), there are two views of NLP: "Symbolic NLP tends to work top-down by imposing known grammatical patterns and meaning associations upon texts. Empirical NLP tends to work bottom-up from the texts themselves, looking for patterns and associations to model, some of which may not correspond to purely syntactic or semantic relationships." Empirical NLP has been widely used since the early 1990's while Symbolic NLP has been viewed less favourably. The system we describe here is in the tradition of Symbolic NLP, as the categories we use have been pre-defined and do not emerge from the data. However, at this point, classification is mainly performed on the basis of formatting structures, not on linguistic constructs.

Our first rule-based classifier used an "if, else" structure to parse the tags returned from Qtag one at a time. The document type was determined solely on the basis of the previous tag and the current tag, and only <Start> tags and the last one or two <End> tags were used to classify a message. A large amount of code (in Python) was written to implement this method, which turned out to be neither efficient nor successful.

Our first approach was too optimistic and too reliant on the document structure given in the SOPs, and our rules did not perform well. Our second implementation of a rule-based classifier uses regular expressions to state the rules. Regular expressions allow us to define more detailed rules and they also allow for more flexibility.

4.1 Regular Expressions as rules

As discussed in (Carr and Estival, 2002), the discrepancy between the format prescribed by the SOPs and the real text input by the operators was one of the main causes of errors. The use of regular expressions as rules allows flexibility in rule definition and result in shorter and more effective code. Several rules can be written to recognise one SOP document type.

The output of Qtag is read into a string. This string contains the list of POS tags for a message. Each rule recognises the tags for one SOP document type and allows any number of other tags in between. Only those POS tags required by the classifier rules are read into the Classifier. Having all the POS tags in a string also allows message headers and multiple SOPs to be pruned off or recognised differently very easily.

We give in (4) an example of a classifier rule for document type "P", where there can be any number of tags before <Frm1> and at least one instance of the separator or more tags before <Frm2>. In (5), we give an example of a message containing a document of type "P".

```
(4)  P = ([ a-za-z]|[ A-Za-za-z0-9]|[ A-ZA-Z]|[ DTG_S]|[
DTG_M]|[ DTG_L]|[ a-za-za-za-za-za-za-z]|[ A-Za-za-za-z]|[
]|){0,}Frm1([ a-za-z]|[ A-Za-za-z0-9]|[ A-ZA-Z]|[ DTG_S]|[
DTG_M]|[ DTG_L]|[ a-za-za-za-za-za-za-z]|[ A-Za-za-za-z]|[
]|){1,}Frm2.([ a-za-z]|[ A-Za-za-z0-9]| [A-ZA-Z]|[ DTG_S]|[
DTG_M]|[ DTG_L]|[ a-za-za-za-za-za-za-z]|[ A-Za-za-za-z]|[
]|){0,}
```

```
(5)  CD DTG_M Frm1 CD From at Frm2 DTG_L
```

4.2 SOP-MRC rules

As mentioned earlier, the classifier rules were first created following the 88 SOP document definitions. They were later derived from a corpus analysis and further refined after analysis of the results on the same corpus. The rules use mostly POS tags relating to formatting, eg. <Frm1> ("formatted 1") or <FrmB> ("formatted B"), but also some content information, with the POS tags for DTGs and <CD> (number). A total of 66 rules were used to recognise the 37 document types that appeared in the VP-02 data. Of these 66 rules, 44 rely on the POS tags for DTGs or <CD>.

One disadvantage of using regular expressions to implement classifier rules might be that they can be fairly long. The example in (4) is one of the shortest rules in terms of number of elements. However, this problem can be alleviated by the use of named groups and the Python interpreter is useful to test the regular expressions before they are included into the classifier.

It is also worth noting that these handcrafted rules were in fact written very quickly, much more quickly than "one rule in two days" as described by (Jackson and Moulinier, 2002).

Each message is tested against all the rules for SOP document types. If no match is found, then the document is assigned to the document type "Free Text". Some rules are in fact subsets of other rules. This defines a hierarchy of rules which can be used to determine the correct SOP document type, see Section 6.

5 Results

We present the results obtained by SOP-MRC on two different message databases. The VP-02 database was used for training the POS Tagger and to define the classifier rules. It contains 2328 messages and 37 document types. The TT-01 database contains 3131 messages and 18 document types. The detailed results for each document type, for both VP-02 and TT-01, are given in Appendix 1 and 2.

The first part of Table 6 shows the overall results of SOP-MRC for VP-02. Since over 75% of the messages are "Free Text", we also show the microaverage result without the "Free Text" category to give a better indication of performance.

The second part of Table 6 shows the overall results of SOP-MRC for TT-01. In this corpus, over 85% of the messages are "Free Text" and the microaverage result is again given without the "Free Text" category.

Recall and Precision Averages	Recall	Precision
VP-02		
Macroaverage	79.99%	67.94%
Microaverage	82.49%	81.52%
Microaverage (no Free Text)	70.53%	43.41%
TT01		
Macroaverage	12.72%	13.09%
Microaverage	86.77%	83.88%
Microaverage (no Free Text)	77.39%	26.81%

Table 7 SOP-MRC for VP-02 and TT01

These results are very encouraging. Although the macroaverage for TT-01 is not very good, this is explained by the fact that there were a number of False Positives for document types which do not occur in this data (see Appendix 2). The microaverage shows that the document types with larger numbers of documents are giving as good results for the new unseen data as for VP-02.

An explanation for the discrepancies between the document types used in VP-02 and in TT-01 is that the SOP definitions were actually developed at DJFHQ, and that VP-02 was a military exercise which only involved that headquarters, with all the messages coming from DJFHQ, while TT-01 was a four nation exercise, with messages coming from a number of different headquarters.

Another issue concerns the "Shift Handover" documents. The Shift Handover form is filled in by officers "handing over" their shift to another officer who "watches" the database for outstanding issues, and is a summary of the past 12 or 24 hours. Although this form is essentially free text, because the officers tend to think in terms of formatted documents, they often write it as another formatted document, eg. with numbered items for new paragraphs. If we classified the Shift Handover form as "Free Text", the accuracy would improve. This can be seen as another example of the well-know fact that the operator or human element is a large factor in system success.

6 Multilabel classification

One of the lessons from our earlier work was that we needed to use a multilabel classification rather than a simple multiclass classification. In multiclass classification, each message is assigned to only one of several possible classes, while in a multilabel classification, a message can be assigned to one or more classes (Lewis, 2002). Our new classifier rules now perform a type of multilabel classification, by assigning a complex label to each message. An example of this complex label is shown in (6).[3]

(6) C:B:A:Free Text

This example shows the output of a message that contains a document of type C. As mentioned above, some rules are actually subsets of other rules, thereby defining a hierarchy of document types. In this case, the rule for document type C includes the rule for document type B, which includes the rule for document type A; thus A and B are also included in the complex label, as well as Free Text, the default classification.

In our current implementation, we choose the label returned by the more specific rule in the complex label, and return it as the single label, or multiclass classification, for the message being classified (in this case, C). Although Sebastiani (2002) argues that a multilabel classifier cannot be used as a single label classifier, the complex label that is returned by our classifier component is in fact a multilabel classification in terms of the hierarchical structure of the classifier rules. This hierarchy can be thought of as a set of binary classifiers (implemented as classifier rules) listed in order, from smallest (more general) to largest (more specific). This set is ordered such that a document type is a subset (in terms of structure, not content) of the next document type.[4]

7 Conclusions and future work

7.1 Improvements

The expanded POS tagset provides a better coverage for the texts in our domain, and the POS Tagger component is now trained for the real data found in military message databases.

The new classifier component is much cleaner and more efficient. Python provides high-level methods to implement regular expressions and use them to search strings of text, which makes it easier to modify the classifier and to add or change the rules.

The results on a new database of messages, which were not used to create the classifier rules, are encouraging and indicate that we can improve the performance of SOP-MRC with little effort.

7.2 Integration with QBI

QBI is an improved search interface to the Lotus Notes operational log database used at DJFHQ, which is developed by the same DSTO team as SOP-MRC. We aim to incorporate the output of SOP-MRC with QBI by providing a category-pivoted view of the documents as categorised by SOP-MRC. An example of this view is shown in Fig. 5.

The categorised view will allow the users of the QBI to quickly find messages in a Lotus Notes operational log database by using document types to limit their search or to locate the relevant message.

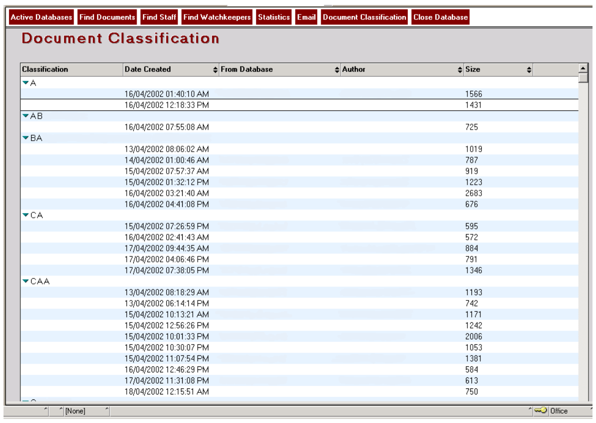
7.3 Other Improvements to SOP-MRC

The performance of the POS Tagger could be improved by pre-processing the messages. Text such as "10 KM" could be normalised to "10KM" so the POS Tagger can properly tag it <DST> rather than <CD> <NN>. This would also help improve the classifier's performance.

The current implementation relies on a one-to-one correspondence between classifier rules and document types. We are looking at another approach, in which a classifier rule would be a subset of a number of rules for a few document types, in other words we would have a more general rule for a set of document types. This would correspond to implementing the true multilabel classification mentioned in Section 6, where the hierarchy of rules also correspond to the conceptual hierarchy of document types.

[3] The names of the document types have been replaced by alphabetical labels for presentation; in the real system, the categories have meaningful labels.

[4] For example, a MEDSITREP (Medical Situation Report) is conceptually a kind of SITREP (Situation Report), but a SITREP is not a kind of "Free Text", even though the rules for "SITREP" and "Free Text" are in a subset relation.



Document Classification

Classification	Date Created	From Database	Author	Size
A				
	16/04/2002 01:40:10 AM			1566
	16/04/2002 12:18:33 PM			1431
AB				
	16/04/2002 07:55:08 AM			725
BA				
	13/04/2002 08:06:02 AM			1019
	14/04/2002 01:00:46 AM			787
	15/04/2002 07:57:37 AM			919
	15/04/2002 01:32:12 PM			1223
	16/04/2002 03:21:40 AM			2683
	16/04/2002 04:41:08 PM			676
CA				
	15/04/2002 07:26:59 PM			595
	16/04/2002 02:41:43 AM			572
	17/04/2002 09:44:35 AM			884
	17/04/2002 04:06:46 PM			791
	17/04/2002 07:38:05 PM			1346
CAA				
	13/04/2002 08:18:29 AM			1193
	13/04/2002 06:14:14 PM			742
	15/04/2002 10:13:21 AM			1171
	15/04/2002 12:56:26 PM			1242
	15/04/2002 10:01:33 PM			2006
	15/04/2002 10:30:07 PM			1053
	15/04/2002 11:07:54 PM			1381
	16/04/2002 12:46:29 PM			584
	17/04/2002 11:31:08 PM			613
	18/04/2002 12:15:51 AM			750

Fig. 5 Screenshot of QBI with SOP-MRC

If an incoming message matches the more general rule, it can then be tested against more specific rules. If the message fails to match one of these, then other methods can be used to determine the more specific document type, but at least the more general category can be kept, rather than defaulting to "Free Text", as is currently the case.

Further analysis to determine the more specific document type would involve the number of certain POS tags, the ordering of these tags or the absence of certain tags.

Another improvement concerns the addition of further POS tags for our domain, for example, tags for unit names and ranks. This information would also be useful in further work on both document classification and information extraction. For instance, it would allow extracting information about which units are involved or identify the personnel injured from NOTICAS messages.

We are also investigating the development of a trainable system, using approaches such as TF-IDF, Rocchio Method, Support Vector Machines or hybrid solutions such as Learning from Positive and Unlabeled text documents (Jackson and Moulinier, 2002; Joachims, 1998; Vapnik, 1995; Lee and Lui, 2003).

References

Carr, O. and Estival, D. (2002) Text Classification of Formatted Text Document. *Proceedings of the 5th Australasian Natural Language Processing Workshop*. Canberra. pp. 49-54.

Jackson, Peter and Moulinier, Isabelle (2002) *Natural Language Processing for Online Applications – Text Retrieval, Extraction and Categorization*, Vol. 5, Natural Language Processing, John Benjamin Publishing Company, Philadelphia.

Joachims, T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the European Conference on Machine Learning*, Springer.

Johnson, D.E., Oles, F.J., Zhang, T., Goetz, T. (2002), A Decision-Tree-Based Symbolic Rule Induction System for Text Categorization, *IBM Systems Journal*, Vol. 41, No. 3, IBM Corporation.

Lee, W.S. and Liu, B. (2003) Learning with Positive and Unlabeled Examples using Weighted Logistic Regression. *Proceedings of the Twentieth International Conference on Machine Learning* (ICML-2003), Washington, DC, USA.

Lewis, David (2002). *Machine Learning for Text Classification Applications*, Tutorial. 40th Meeting of the Association of Computational Linguistics, University of Pennsylvania. Philadelphia, USA.

Manning, C.D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts.

Mason, Oliver (2003). *Qtag 3.1*. Department of English, School of Humanities, University of Birmingham, http://web.bham.ac.uk/O.Mason/software/tagger/

Sebastiani, Fabrizio. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, Vol. 34, No. 1, pp. 1-47.

Sutcliffe, Richard, Koch, Heinz-Detlev and McElligott, Anne (eds) (1996), *Industrial Parsing of Software Manuals*. Amsterdam, Rodopi.

Tufis, Dan and Mason, Oliver (1998). Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger, *Proceedings of the First International Conference on Language Resources & Evaluation (LREC)*, Granada, Spain, 28-30 May 1998, p.589-596.

Vapnik, V. N. (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Doc Type	Gold	Total	TP	FP	FN	Recall	Precision
A	1	2	1	1	0	100.00%	50.00%
B	1	1	1	0	0	100.00%	100.00%
C	10	6	6	0	4	60.00%	100.00%
D	2	5	1	4	1	50.00%	20.00%
E	14	10	3	7	11	21.43%	30.00%
F	1	2	1	1	0	100.00%	50.00%
G	6	5	4	1	2	66.67%	80.00%
H	1	1	1	0	0	100.00%	100.00%
I	2	2	2	0	0	100.00%	100.00%
J	1	3	1	2	0	100.00%	33.33%
K	11	18	7	11	4	63.64%	38.89%
L	33	16	9	7	24	27.27%	56.25%
Free Text	1846	1657	1625	32	221	88.03%	98.07%
M	3	2	2	0	1	66.67%	100.00%
N	19	28	8	20	11	42.11%	28.57%
O	2	4	2	2	0	100.00%	50.00%
P	164	391	131	260	33	79.88%	33.50%
Q	34	22	13	9	21	38.24%	59.09%
R	3	3	3	0	0	100.00%	100.00%
S	32	27	27	0	5	84.38%	100.00%
T	1	4	1	3	0	100.00%	25.00%
U	20	30	18	12	2	90.00%	60.00%
V	5	5	5	0	0	100.00%	100.00%
W	1	3	1	2	0	100.00%	33.33%
X	1	2	1	1	0	100.00%	50.00%
Y	5	2	2	0	3	40.00%	100.00%
Z	1	11	1	10	0	100.00%	9.09%
AA	1	1	1	0	0	100.00%	100.00%
BB	2	3	2	1	0	100.00%	66.67%
CC	1	1	1	0	0	100.00%	100.00%
DD	81	31	31	0	50	38.27%	100.00%
EE	3	4	2	2	1	66.67%	50.00%
FF	1	2	1	1	0	100.00%	50.00%
GG	1	1	1	0	0	100.00%	100.00%
HH	1	1	1	0	0	100.00%	100.00%
II	13	19	8	11	5	61.54%	42.11%
JJ	4	3	3	0	1	75.00%	100.00%
Total	2328	2328					

Appendix 1 SOP-MRC on VP02

Doc Type	Gold	Total	TP	FP	FN	Recall	Precision
A	0	1	0	1	0	0.00%	0.00%
B	0	1	0	1	0	0.00%	0.00%
C	1	0	0	0	1	0.00%	0.00%
D	2	0	0	0	2	0.00%	0.00%
E	10	11	0	11	10	0.00%	0.00%
F	0	1	0	1	0	0.00%	0.00%
G	0	1	0	1	0	0.00%	0.00%
J	0	2	0	2	0	0.00%	0.00%
K	0	14	0	14	0	0.00%	0.00%
L	26	10	4	6	22	15.38%	40.00%
Free Text	2767	2493	2471	22	296	89.30%	99.12%
M	0	1	0	1	0	0.00%	0.00%
N	5	26	3	23	2	60.00%	11.54%
O	2	2	2	0	0	100.00%	100.00%
P	176	521	167	354	9	94.89%	32.05%
Q	57	0	0	0	57	0.00%	0.00%
S	5	1	0	1	5	0.00%	0.00%
T	0	3	0	3	0	0.00%	0.00%
U	0	1	0	1	0	0.00%	0.00%
V	1	0	0	0	1	0.00%	0.00%
W	0	3	0	3	0	0.00%	0.00%
Y	0	1	0	1	0	0.00%	0.00%
Z	0	7	0	7	0	0.00%	0.00%
AA	14	0	0	0	14	0.00%	0.00%
CC	1	0	0	0	1	0.00%	0.00%
DD	50	10	1	9	49	2.00%	10.00%
EE	2	16	0	16	2	0.00%	0.00%
FF	1	0	0	0	1	0.00%	0.00%
II	3	1	0	1	3	0.00%	0.00%
JJ	5	1	1	0	4	20.00%	100.00%
Total	3128	3128					

Appendix 2 SOP-MRC on TT01

Identifying Interpersonal Distance using Systemic Features

Maria Herke-Couchman and **Casey Whitelaw**
Language Technology Research Group
Capital Markets Co-operative Research Centre
University of Sydney
{*maria, casey*}*@it.usyd.edu.au*

Abstract

This paper uses Systemic Functional Linguistic (SFL) theory as a basis for extracting semantic features of documents. We focus on the pronominal and determination system and the role it plays in constructing interpersonal distance. By using a hierarchical system model that represents the author's language choices, it is possible to construct a rich and informative feature representation. Using these systemic features, we report clear separation between registers with different interpersonal distance.

1 Introduction

This paper explores the categorisation of text based on meaning. Rather than classify on the content matter of a document, we aim to capture elements of the manner in which the document is written. In particular, we use a computational model of Systemic Functional Linguistic theory to identify the interpersonal distance of a text.

Previous work has looked at extracting other semantic properties of documents. This has included the subjectivity or objectivity of whole texts (Kessler et al., 1997) or individual sentences (Wiebe, 1990) (Riloff et al., 2003), and classifying reviews as positive or negative (Turney, 2002). Here, we investigate the interpersonal distance, which partially describes the type of relationship established between author and reader.

Much of the prior research has focused on semantic categories of adjectives (Turney, 2002) and nouns (Riloff et al., 2003). This paper focuses on the closed class of pronominals and determiners. While the use of these individual words may provide some semantic information, it is through placing them in a system of language choice that patterns of usage may be correlated with interpersonal distance.

Systemic Functional theory is a linguistic theory that interprets a text as an exchange of meanings and, while these meanings are realised by grammar and words, a text, in the first instance, is viewed as a semantic unit (Halliday and Hasan, 1985). In a study that seeks to categorise a text according to the meanings that it makes rather than just the words that it uses, SF theory presents itself as an extremely useful model. We propose preliminary methods for computing aspects of Systemic Functional Linguistics at the lexical level, without dependence on semantic resources or parsers. We show that SFL is well-suited to identifying document-level characteristics of language use.

2 Systemic Functional Linguistics

Systemic Functional Linguistics (SFL) is a framework for describing and modeling language in functional rather than formal terms. The theory is *functional* in that language is interpreted as a resource for making meaning, and descriptions are based on extensive analyses of written and spoken text (Halliday, 1994). The theory is also *systemic* in that it models language as a system of choices (Matthiessen, 1995). SFL has been applied in natural language processing in various contexts since the 1960s, but has been used most widely in text generation (Matthiessen and Bateman, 1991) (Teich, 1995).

SF theory describes the use of language in context. It is conceptualised as a multi-dimensional semiotic space showing the organisation of language, both globally as a meaning making system and locally as sub-systems of language use.

This paper deals with interpersonal distance, which is an aspect of the way in which the author establishes a dialogue with the reader. To understand the function of this system, we show its place within the interpersonal metafunction, and how this is realised within the semantic and lexicogrammatical strata.

2.1 The Interpersonal Metafunction

The metafunctions refer to the three separate strands of meaning that contribute to the overall meaning in the text (Halliday, 1994). These three metafunctions are deployed simultaneously and are the textual, the interpersonal and the ideational:

- The textual metafunction provides 'the resources for presenting information as text in context' (Matthiessen, 1995)
- The interpersonal metafunction provides the resources for enacting social roles and relations as meaning.
- The ideational metafunction provides the resources for construing our experience of the world.

The phenomenon we are exploring, interpersonal distance, is located within the interpersonal metafunction and relates to the tenor of the relationship between the writer and reader within the context.

2.2 Realisations of Meaning

One key global dimension is the hierarchy of stratification. Language itself is modelled as an ordered series of levels or strata, as shown in Figure 1. Interpersonal distance expresses an aspect of the meaning of the text, and so is located

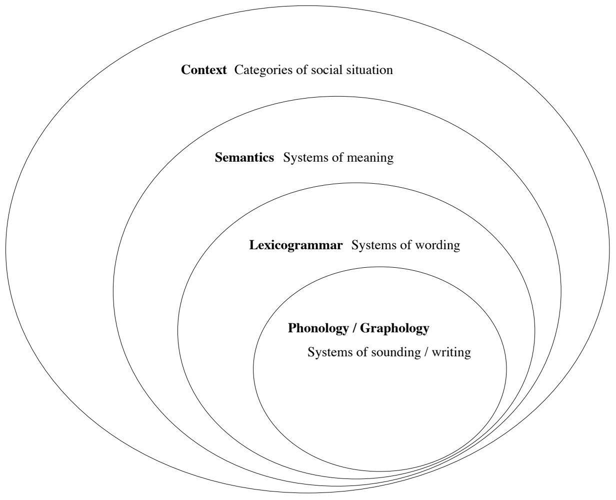

Context Categories of social situation
Semantics Systems of meaning
Lexicogrammar Systems of wording
Phonology / Graphology
Systems of sounding / writing

Figure 1: Modelling language stratally (Hasan, 1996)

within the semantic statum. As a pattern of meaning, interpersonal distance is realised as a pattern of wording in the lexicogrammar. That is, the meaning is expressed through a pattern of word usage.

2.3 Characterising Registers

A register is a group of texts whose language selections vary from the general language system in similar ways. A register can be characterised by properties of its field, tenor, and mode. Registers are skewings 'of probabilities relative to the general systemic probabilities' (Matthiessen, 1993). Register is the instantiation of particular situation types within the system.

While a register groups documents on the basis of the meanings they make, these meanings are realised in the semantics and lexicogrammar of the texts, and so may be analysed on these terms. In particular, registerial differences should be exposed through the patterns of language choice within a system.

2.4 Interpersonal Distance

Interpersonal distance refers to the distance between speaker and addressee (Eggins et al., 1993). Typically, spoken discourse that includes oral and visual contact is representative of minimal interpersonal distance whereas written discourse with no visual, oral or aural contact rep-

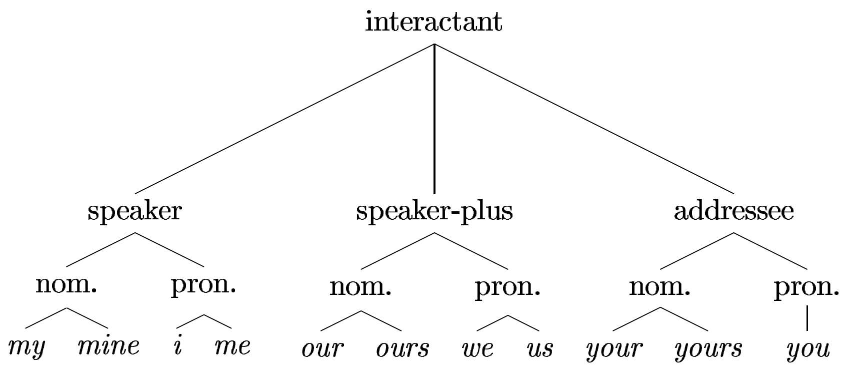


Figure 2: interactant

resents maximal interpersonal distance.

Interpersonal distance can be determined by analysing various systemic language choices made within a text. Examples of such an analysis might include measuring the degree and frequency of participant nominalisation deployed within a text as well as the frequency and type of interactant reference (Eggins et al., 1993) (Couchman, 2001).

An example of a text with very close interpersonal distance would be one that includes direct speech, such as the following (Biggs, 1990):

> Kupe went to Muturangi's village and spoke of the bad behaviour of the animal with regard to his people's bait, saying, '**I** have come to tell **you** to kill **your** octopus.', Muturangi replied, '**I** won't agree to **my** pet being killed. Its home is in the sea.' 'Well', said Kupe, 'if **you** won't take care of **your** pet, **I** will kill it.' Kupe went back home and said to his people, 'Prepare **my** canoe as well.' Maataa-hoorua was made ready and Kupe set off to go.

In the above text, degree and frequency of nominalisation is low and selections from the Interactant system (bold face) are high.

A written history text, is a good example of a text that constructs maximum interpersonal distance partly by making no selections from within the Interactant system (Biggs, 1997):

> The discovery of Hawaii from the Marquesas was a remarkable achievement, but at twenty degrees north latitude Hawaii is still within the zone of the trade winds that blow steadily and predictably for half of each year. New Zealand lies far to the South of the trade winds, in the stormy waters and unpredictable weather of the Tasman Sea. The Southern hemisphere, moreover, has no Pole Star to provide a constant compass point.

Work on Nigerian emails has indicated that close interpersonal distance might be characteristic of that particular register (Herke-Couchman, 2003). As mentioned above, interpersonal distance can be analysed through the various systemic language choices. In this paper, we will explore the Interpersonal distance established within a text by analysing just the pronominal and determination system (Matthiessen, 1995). This system encapsulates the Interactant system.

One possible way of measuring the distance between speaker and addressee lexicogrammatically is to explore the language choices made within the pronominal and determination system (Matthiessen, 1995).

2.5 The Pronominal & Determination System

The Pronominal and Determination system is a language system that includes within it the interpersonal resource for modelling the relationship between the interactants in the dialogue. The system is a closed grammatical system that includes realisations of both interactant (speaker, speaker-plus and addressee) and non-interactant reference items.

It is expected that very close interpersonal distance in a text would be characterised by frequent selections from the interactant systems. For example, a text seeking to establish patterns of familiarity between author and reader would show foregrounded patterns of speaker (I, me, my, mine) and addressee (you, your, yours) usage. Contrastively, a text that is constructing a more formal and distant tenor will typically make little use of the interactant systems but may instead show strong patterns of usage of more generalised alternative meaning systems.

3 Representing System Networks

For systemic information to be extracted from a document, there must be a suitable computationally-feasible language model. While SFL is a comprehensive and multidimensional linguistic theory, and is not obviously computationally tractable, we can develop a more restricted model that allows us to work with specific systems such as determination.

As is shown in Figures 2 and 3, this system can intuitively be modelled as a tree. Each internal node represents a subsystem or category: a pattern of possible language choice. Each leaf gives a realisation of its parent system as a word or phrase. A system may contain both lexical realisations and subsystems, as in the 'singular-conscious-female' example in Figure 3.

This is an impoverished but still useful view of a system network. Language choice does not always result in a specific word or phrase; an in-depth manual analysis of a text would show that grammatical and lexical units of various sizes contribute to the overall meaning. Further, interaction between systems can result in networks that are not strictly heirarchical, and richer representations will be required to model these processes effectively. The current representation is sufficient to capture language choice for a system such as determination, which is a closed class and fully lexically realised.

The usage of a system in a document can be represented by a system instance. Each occurrence of each lexical realisation in the document is counted, and these counts are accumulated upwards through the network. The count at an internal node is the sum of the counts of its subcategories. This process is no more costly than constructing a feature vector in traditional text classification methods.

In a standard 'bag-of-words' approach, the contribution of a word to a document is given by its relative frequency: how rarely or often that word is used. This implicitly uses a language model in which all words are independent of each other. Crucially, this does not and cannot take into account the *choice* between words, since there is no representation of this choice. Placing words within a system network provides a basis for richer and more informative feature representation.

There are two main advantages gained by adding systemic information for feature representation. Firstly, it allows for categorical features that are based on semantically-related groups of words, at all levels in the network. By collecting aggregate counts, individual variations within a category are ignored. For a given register, it may be the case that important and characteristic language choice occurs at a very fine level, distinguishing between usage of individual words. This word-level information is kept intact, as in a bag-of-words approach. In another register, it may be the usage of a category, such as interactant, that is characteristic. The usage of any words within the category may appear random while maintaining consistent category usage. These higher-level features are not available in a traditional bag-of-words approach, in which these patterns may be lost as noise.

The second and more important difference to traditional feature representation is the representation of language choice. Not only can a system instance calculate the frequency of usage for categories within a system, it can calculate the relative usage within a category. *System contribution* is simply the ratio of subcategory occurence count to super-category occurence count, or a normalisation across ele-

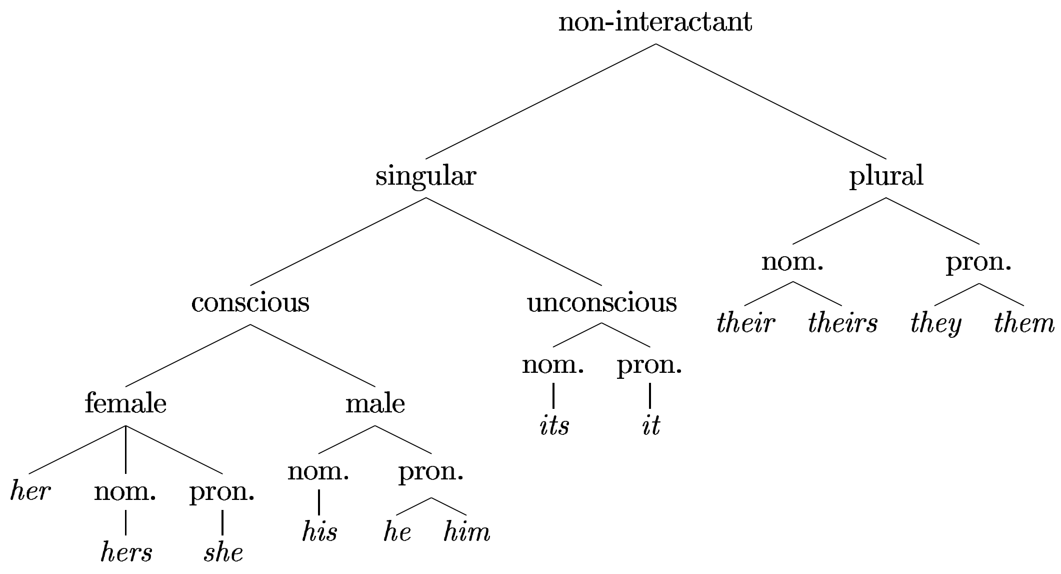


Figure 3: non-interactant

ments within a category. This gives rise to features such as 'interactant usage versus non-interactant usage'. This directly models the fact that in using language, a choice is made. It is a choice not between one word and any other (choosing between unrelated words such as 'dog' and 'elegant'), but between semantic categories within a system. Comparative features such as these can only be used together with a sensible basis for comparison, which is provided here through the use of SFL.

System contribution is not proportional or strongly correlated to term frequency, and the two measures provide useful and complementary information. Term frequency reports the percentage of a document that is made up of a given term. Within a system instance, term frequency can be used to report the term frequency not just of terms but of systems as well. Unlike term frequency, system contribution does not capture how often a system is used, but rather its usage in relation to the other possible choices. In the same way as a register may be characterised by choice, it may also be characterised by frequent usage of a particular system. This gives two complementary representations that may each be useful in discerning interpersonal distance.

4 Identifying Registers

As discussed in Section 2.3, a register is constrained in the types of meanings it is likely to construct. A register may be characterised as establishing a certain interpersonal distance. If the choice within the determination system reflects this semantic position, we should be able to classify documents on this basis.

Not all registers are distinguishable by interpersonal distance. This is but one of many of the semantic properties that characterise documents, such as formality, modality, and evaluation. Note also that the identification of a register is not the same as identifying the *topic* of a document; instances of the 'newspaper article' register may have very different content that is presented in the same fashion.

4.1 Corpora

We chose corpora that were clearly separated into different registers. From prior manual analysis, it was expected that these registers would have different characteristic interpersonal distance.

Previous work has examined the use of the determination system in so-called 'Nigerian emails'. These are fraudulent emails in which the author attempts to establish an illegal business relationship (money transfer) with the recipient. One of the most salient characteristics of this register is the way in which the author, despite having no prior relationship with the reader, works to set up a sense of familiarity and trust. These semantic strategies suggest closer interpersonal distance than would usually be expected in the setting up of a legitimate business relationship, particularly since the texts are written rather than spoken. This corpus contained 67 manually collected Nigerian emails.

The Nigerian emails were contrasted with a collection of newspaper articles taken from the standard Reuters text classification corpus. Since many of the newswire texts are very short, only texts with more than one thousand words were kept, resulting in 683 documents. As a result of the context in which they unfold, it was expected that the Reuters newswire texts would make different language choices in order to realise the different meanings they construct. More specifically, it is expected that this register constructs greater interpersonal distance between author and reader.

The third register was taken from the British National Corpus and consists of 195 documents marked as belonging to the 'spoken / leisure' category. These are mostly transcriptions of interviews and radio shows covering a wide range of topics. As stated above, the interpersonal distance constructed in spoken text is almost always much closer than that constructed in written texts. Including this corpus allowed us to explore whether the perceived close interpersonal distance in the Nigerian email corpus would be confused with the close interpersonal distance that is typical of spoken texts.

These corpora differ greatly in both field and tenor, and can be separated easily using standard bag-of-words techniques. In using these corpora, we aim not to show improved performance, but to show that the determination system provides sufficient evidence to separate documents on the basis of interpersonal distance. For this to be possible, the words and categories in this system must be used in a regular and learnable fashion, which reflects the semantic positioning of the text.

4.2 Features Used

The behaviour of a system within a document can be represented as a system instance. As discussed in Section 3, a system instance stores heirarchical information at every level from full system to individual lexical realisations. System usage may differ at any or all of these levels: some registers may make very specific lexical choices, while others may be differentiable by more general trends. In its entirety, the determination system consists of 109 nodes including 48 lexical realisations. From these, various subsets were used to test the performance and robustness of the system.

all All 109 system and lexis nodes.

lexis The 48 lexical realisations in the system.

system All 61 non-lexical features.

top10 Top 10 features on the basis of information gain

top5 Top 5 features on the basis of information gain

Each set of features was computed once using term frequency (percentage of document) and again using system contribution (percentage of supersystem). Classification was performed using three different machine learners, all commonly used in text classification tasks: a Naive Bayes probabilistic classifier (NB), a decision tree (J48), and a support vector machine (SVM). All implementations are part of the publicly available WEKA machine learning package (Witten and Eibe, 1999).

4.3 Results

Results from using system contribution and term frequency are shown in Tables 1 and 2 respectively. All of the feature sets and classifiers produced clear separation of the classes, using only features from the determination system. The best result of 99.6% came from the use of an SVM use the system contribution data of either all features or lexical features. It is clear from these results that these corpora are

	#atts	NB	J48	SVM
all	109	**99.4%**	97.9%	**99.6%**
lexis	48	98.6%	**98.6%**	**99.6%**
system	61	98.6%	98.1%	99.5%
top10	10	98.9%	97.7%	98.6%
top5	5	96.2%	98.1%	98.2%

Table 1: classification accuracy using system contribution

	#atts	NB	J48	SVM
all	109	92.8%	98.2%	98.3%
lexis	48	93.8%	98.1%	**98.4%**
system	61	93,9%	98.4%	98.3%
top10	10	96.1%	**98.6%**	97.9%
top5	5	**97.3%**	98.1%	97.8%

Table 2: classification accuracy using document percentage

separable using features related to interpersonal distance.

Better results were achieved using system contribution than term frequency. By measuring the system choice, rather than system usage, this feature representation highlights the salient aspects of language use. This contrastive description is made possible by placing words in a system network.

In all tests, the Nigerian and Reuters corpora were clearly separated. These registers have markedly different and strongly characteristic interpersonal distance. The spoken corpus exhibited a small amount of confusion with the Nigerian texts, showing evidence that their language is more like spoken than written text.

Feature selection exhibits different effects on the two types of features used. Best performance for system contribution features came from using all features, or only lexical features. Best performance for term frequency features, however, came from using fewer features. Since there is a high degree of correlation between term frequencies within a system network, this can skew results when using classifiers that assume indendent features, as Naive Bayes does.

Table 3 shows the top 10 features as ranked using the information gain metric (Quinlan, 1993). For systemic features, almost all are located within the *interactant* subsystem (Figure 2. This is further confirmation that the discerning features are not random discrepancies between classes, but are evidence of the underlying semantic intent. Also shown are the most significant features in the bag-of-words approach. Despite having access to all the words in the documents, the most significant were still those located in the determination system, together with transcribed discourse markers such as 'er' and 'erm'.

5 Conclusion

SFL is fundamentally a theory of meaning. As such, language choices can be identified as both formal lexical or grammatical selections as well as in terms of systemic meaning selections. The relationship between these two complementary perspectives is one of abstraction or generalisation; a meaning system is more abstract than the grammar or lexis that realises it (Martin and Rose, 2003). This realisation ensures that a meaning phenomenon such as interpersonal distance is characterisable in terms of both systemic choice and lexicogrammatical structure.

In this paper, we have shown that one aspect of the interpersonal distance of a document can be characterised by the use of the determination system. We have further shown that registers that construct variable interpersonal meaning can be separated solely using the features from the Pronominal and Determination system. This can be achieved by modelling SFL at the lexical level without specific external resources.

Interpersonal distance is but one property of the tenor of a document. Similarly, the determination system is but one small part of SFL theory. As our ability to computationally model and extract system networks increases, these systems and their interactions will provide more features by which the semantic properties of a document may be discerned.

system	term	bag-of-words
addressee / pronominal	addressee / nominal	*your*
addressee / nominal	*your*	*my*
speaker / pronominal	*my*	*you*
speaker / nominal	interactant	*me*
I	addressee	*said*
interactant	speaker / nominal	*I*
non-interactant	addressee / pronominal	*er*
your	*you*	*that's*
me	*I*	*erm*
my	speaker / pronominal	*us*

Table 3: Top ten features from each method of representation

References

Bruce Biggs. 1990. *In the Beginning, The Oxford Illustrated History of New Zealand*. Oxford University Press, Oxford.

Bruce Biggs. 1997. *He Whirwhiringa Selected Redaings in Maori*. Auckland University Press, Auckland.

Maria Couchman. 2001. Transposing culture: A tristratal exploration of the meaning making of two cultures. Master's thesis, Macquarie University.

S. Eggins, P. Wignell, and J. R. Martin, 1993. *Register analysis: theory and practice*, chapter The discourse of history: distancing the recoverable past, pages 75–109. Pinter, London.

Michael A. K. Halliday. 1994. *Introduction to Functional Grammar*. Edward Arnold, second edition.

R. Hasan. 1996. *Ways of saying, ways of meaning: selected papers of Ruqaiya Hasan*. Cassell, London.

M. A. Herke-Couchman. 2003. Arresting the scams: Using systemic functional theory to solve a hi-tech social problem. In *ASFLA03*.

Brett Kessler, Geoffrey Nunberg, and Hinrich Schütze. 1997. Automatic detection of text genre. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 32–38, Somerset, New Jersey. Association for Computational Linguistics.

J. R. Martin and David Rose. 2003. *Working with Discourse: Meaning Beyond the Clause*. Continuum, London and New York.

C. M. I. M. Matthiessen and J. A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Frances Pinter Publishers and St. Martin's Press, London and New York.

C. M. I. M. Matthiessen, 1993. *Register analysis: theory and practice*, chapter Register in the round: diversity in a unified theory of register, pages 221–292. Pinter, London.

Christian Matthiessen. 1995. *Lexico-grammatical cartography: English systems*. International Language Sciences Publishers.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of CoNLL-2003*, pages 25–32. Edmonton, Canada.

Elke Teich. 1995. *A Proposal for Dependency in Systemic Functional Grammar – Metasemiosis in Computational Systemic Functional Linguistics*. Ph.D. thesis, University of the Saarland and GMD/IPSI, Darmstadt.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424, Philadelphia, Pennsylvania.

J. Wiebe. 1990. *Recognizing Subjective Sentences: A Computational Investigation of Narrative Text*. Ph.D. thesis, State University of New York at Buffalo.

Ian H. Witten and Frank Eibe. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.