Australasian Language Technology Association Workshop 2008

Proceedings of the Workshop

Workshop Chairs:
Nicola Stokes
David Powers

8-10 December 2008
CSIRO ICT Centre
Hobart, Australia

Proceedings of the Australasian Language Technology Association Workshop 2008 (ALTA 2008)

URL: http://www.alta.asn.au/events/alta2008/

Sponsors:

ISSN 1834-7037 (online proceedings)

Preface

This volume contains the papers accepted for presentation at this year’s Australasian Language Technology Workshop (ALTA2008), held at the CSIRO Tasmanian ICT Centre, Hobart, Australia, on December 8-10, 2008. This is the sixth annual installment of the workshop since the formation of ALTA, and the continuation of an annual workshop series that existed under various guises throughout the 1990s.

The goals of the workshop are:

- to bring together the Language Technology (LT) community in Australasia;
- to encourage interactions with the international LT community;
- to foster interaction between academic and industrial researchers;
- to encourage dissemination and peer evaluation of research results;
- to provide a forum for the discussion of new and ongoing research and projects;
- to allow the broader AI community to become aware of local LT research;
- to increase visibility of LT research in Australia, New Zealand and overseas;
- to promote, support and encourage the participation of students in LT research.

Once again, this year ALTA is co-located with the Australasian Document Computing Symposium (ADCS), including a joint session of papers and talks of interest to both communities. This year’s Australasian Language Technology Workshop includes regular talks as well as poster presentations which rank equally as papers. The number of papers submitted increased by 52% compared to last year, and 20 papers were selected by formal review to appear in these proceedings. This represents an increase of just 25% in the number of papers accepted, and acceptance rate is thus 57% as compared to 70% last year. Ten of the submissions came from outside Australasia, and two submissions were accepted from Indonesia.

Of the 20 accepted papers, 14 are oral presentations and 6 are poster presentations. Each full-length submission was independently peer reviewed by at least two members of the international program committee, in accordance with the DEST requirements for E1 conference publications.
The conference is classified as category B in the Computing Research and Education Association of Australasia (CORE) classification system which will form the basis of ranking computer science conference publications for ARC.

We would like to thank all the authors who submitted papers, the members of the program committee for the time and effort they contributed in reviewing the papers; Rosie Jones (Yahoo!) for joining us as Keynote Speaker, as well as ADCS for their joint support of the conference and keynotes. Our thanks also go to local organizers Dipak Bhandari and Shlomo Berkovsky (CSIRO), to webmasters Darius Pfitzner and Richard Leibbrandt (Flinders University) for their logistical support, to members of the ALTA executive for their assistance in organizing the workshop, and to our sponsors (NICTA and CSIRO) who enabled us in particular to support student participation and accommodation, as well as free registration for all early registrants.

David Powers and Nicola Stokes
Programme Co-Chairs

Programme

Monday December 8, 2008
ALTA/ADCS Joint Session
3:45pm-5:00pm: (4 papers, 1 keynote)

- Sumukh Ghodke and Steven Bird, "Querying linguistic annotations" (ADCS)
- Karl Grieser, Timothy Baldwin, Fabian Bohnert, and Liz Sonenberg, "Using collaboratively constructed document collections to simulate real world object comparisons" (ADCS)
- Katie Bell and James R Curran, "Answer Attenuation in Question Answering" (ALTA)
- Clint Burfoot, "Using multiple sources of agreement information for sentiment classification of political transcripts" (ALTA)

5:00pm-6:00pm: Joint ADCS/ALTA Keynote

- "Syntactic and Semantic Structure in Web Search Queries"
  Rosie Jones, Yahoo!

6:00pm-7:00pm: Canapes and Drinks (Incorporating ADCS Best Paper Awards)
7:00pm-10:00pm Conference Dinner (not covered by registration)

Tuesday December 9, 2008
Session 1
9:30-10:30am: (2 papers)

- Andrew Lampert, Robert Dale and Cecile Paris, "Requests and Commitments in Email are More Complex Than You Think: Eight Reasons to be Cautious"
- Olivia March and Timothy Baldwin, "Automatic Event Reference Identification for Text Summarisation"

10:30am-11:00am: Coffee break

Session 2
11:00am-12:00pm: (2 papers)

- Susan Howlett and James Curran, "Automatic Acquisition of Training Data for Statistical Parsers"
- Tara McIntosh and James R Curran, "Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition"

Poster Sessions (posters should be mounted by the end of morning tea time)
12:00pm-12:30pm: Speed papers I (3 speed papers = 30 mins)

- Femphy Pisceldo, Rahmad Mahendra, Ruli Manurung and I Wayan Arka, "A Two-Level Morphological Analyser for the Indonesian Language"
- Michael Fridkin and David Dowe, "Lexical Access via Phoneme to Grapheme conversion"

- Zhihan Li, Yue Xu and Shlomo Geva, "Text Mining Based Query Expansion for Chinese IR"

12:30pm-2:00pm: Lunch (posters may be previewed)
2:00pm-2:30pm: Speed papers II (3 speed papers =30 mins)

- Jonathan K Kummerfeld and James R Curran, "Classification of Verb Particle Constructions with the Google Web1T Corpus"
- Simon Zwarts and Mark Dras, "Morphosyntactic Target Language Matching in Statistical Machine Translation"
- Daniel Tse and James Curran, "Punctuation normalisation for cleaner treebanks and parsers"

2:30pm-3:30pm: - Posters
3:30pm-4:00pm: Coffee Break (poster session can continue)

Session 3
4:00pm-5:00pm: (2 papers)

- Jette Viethen and Robert Dale, "Generating Relational References: What Makes a Difference?"
- Eliza Margaretha and Ruli Manurung "Comparing the value of Latent Semantic Analysis on two English-to-Indonesian lexical mapping tasks: What is it good for?"

Wednesday December 10, 2008 (5 papers)
Session 4
9:30am-11:00: (3 papers)

- Dominick Ng, David J Kedziora, Terry T W Miu and James R Curran, "Investigating Features for Classifying Noun Relations"
- Joel Nothman, James Curran and Tara Murphy, "Transforming Wikipedia into Named Entity Training Data"
- Jeremy Nicholson and Timothy Baldwin, "Learning Count Classifier Preferences of Malay Nouns"

11:00-11:30am: Coffee break

Session 5
11:30am-1:00pm: (3 papers)

- Cecile Paris, Nathalie Colineau, Andrew Lampert and Joan Giralt Duran, "Fit it in but say it well!"
- Tobias Kuhn and Rolf Schwitter, "Writing Support for Controlled Natural Languages"
- Ari Chanen and Jon Patrick, "All-Topology, Semi-Abstract Syntactic Features for Text Categorization"

ALTA Best Paper & Best Presentation Awards
1:00pm-2:00pm: Lunch (incorporating ALTA Best Paper Awards)
2:00pm-2:45pm: ALTA Annual General Meeting

Table of Contents

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 1-1

Keynote Address: Syntactic and Semantic Structure in Web Search Queries

Dr Rosie Jones, Yahoo!

Abstract: Traditionally, information retrieval examines the search query in isolation: a query is used to retrieve documents, and the relevance of the documents returned is evaluated in relation to that query. The query itself is assumed to consist of a bag of words, without any grammatical structure. However, queries can also be shown to exhibit grammatical structure, often consisting of telegraphic noun-phrases. In addition, users typically conduct web and other types of searches in sessions, issuing a query, examining results, and then re-issuing a modified query to improve the results. We describe the properties of real web search sessions, and show that users conduct searches for both broad and finer grained tasks, which can be both interleaved and nested. Reformulations reflect many relationships, including synonymy, hypernymy and hyponomy. We show that user search reformulations can be mined to identify related terms, and that we can identify the boundaries between tasks with greater accuracy than previous methods.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 2-10

Answer Attenutation in Question Answering

Katie Bell and **James R. Curran**
School of Information Technology
University of Sydney
Sydney, Australia
{kbel5892, james}@it.usyd.edu.au

Abstract

Research in Question Answering (QA) has been dominated by the TREC methodology of black-box system evaluation. This makes it difficult to evaluate the effectiveness of individual components and requires human involvement. We have collected a set of answer locations within the AQUAINT corpus for a sample of TREC questions, in doing so we also analyse the ability of humans to retrieve answers. Our answer corpus allows us to track answer attenuation through a QA system. We use this method to evaluate the Pronto QA system (Bos et al., 2007).

1 Introduction

A Question Answering system, is any system which answers questions posed in natural language. In its earliest forms, QA systems were natural language front-ends to structured databases of knowledge (Androutsopoulos, 1995). Today, there exists a massive quantity of data freely available on the internet in raw textual form, but to find specific information the user may be required to read many documents returned from a query. The goal of open domain QA is to enable users to find specific answers to questions from enormous corpora spanning many domains. QA can be seen as a search problem: finding answers in a corpus of text. A QA system reduces the search space in stages, starting with selecting documents, then passages, and so on, until a single answer is returned.

The current method of evaluating Question Answering systems stems from the Text REtrieval Conference (TREC) Question Answering track. Using a standard document collection and question set, each participating system is graded on the proportion of answers which are correct and supported by the source document. These criteria are assessed by the only reliable way to determine the correctness and support of an answer: humans. However, manual evaluation is costly and not feasible for constant evaluation of QA while developing new systems or techniques. For this purpose a set of correct answer regular expressions are crafted from the correct answers as judged by the TREC QA human assessors. These answer keys are unreliable as they are both not exhaustive and do not take into account the support of the source document (Lin, 2005).

For reliable automated evaluation we need a gold standard dataset. This would be an exhaustive list of the exact locations of human verified correct answer instances for a set of questions. Using such a set of answer locations, we can define an extension of current evaluation practices, which we call *answer attenuation*. Answer attenuation is the proportion of correct answers lost from the search space after each stage of processing. The purpose of this measure is both to provide more information about the effectiveness of each system component and a more reliable overall system evaluation.

With group of volunteer annotators, we have collected an initial test set of answer locations within a document collection which is an approximation of this theoretical gold standard. From analysing the annotators' behaviour we find that on an individual scale, humans are not adept at exhaustively finding answers. We used this data to analyse the answer attenuation of the Pronto QA system (Bos et al., 2007). In doing so, we revealed some weaknesses both in Pronto and in the use of TREC answer keys as an evaluation measure.

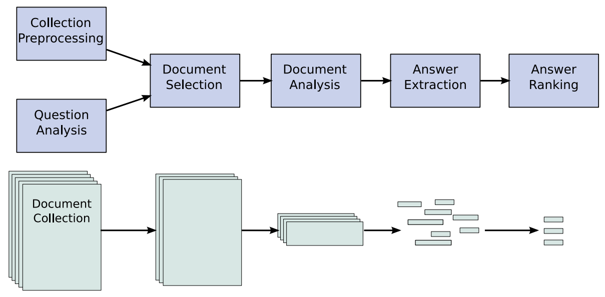


Figure 1: The generic structure of QA systems

2 Background

Question answering can be seen as a search problem. QA systems typically follow a pattern of successively narrowing the search space for answers. The generic structure defined by Hirschman and Gaizauskas (2001) provides a model for describing QA systems. Not all QA systems include all of these components, yet it provides a useful framework to draw parallels. The generic system structure is shown in Figure 1, starting with pre-processing of the document collection, usually indexing to improve retrieval time. The question analysis stage involves determining the expected answer type of the question and generating keywords to be used in retrieving documents in the document selection stage. The document analysis stage involves selecting sections from the text which are considered likely to contain answers. Short phrases which match the expect answer type of the question are selected in the answer extraction stage, these answers are then ranked according to how well they match or answer the original question.

2.1 QA Evaluation

The methods for evaluating and analysing QA systems can be divided into three categories: black-box whole system evaluation, component specific evaluation and system-wide component analysis.

Black-box Evaluation The results of a human assessment of an answer can be approximated using human generated answer keys (Breck et al., 2000). This method remains the most widely used form of evaluation of QA systems. However, this method should be used with caution (Lin, 2005). Lin's work calls into question the reliability of the use of these answer keys for system evaluation and comparison. Firstly, the answer keys do not take into account the support of the source document. Similarly, the answers are limited to those returned by existing systems. Any new QA system will not be evaluated as better than the sum of preceding systems. Consequently, Lin finds that these evaluation resources underestimate answer accuracy.

Lin proposes a potential solution as future work which is very similar to the method proposed here: the tagging of all instances of the correct answer for each question. He then highlights the difficulties of such an approach, particularly in the creation of the test data. There is no efficient way to exhaustively find all answer instances without a manual search of the entire document collection. Searching for the known answer string is unreliable as answers can appear in multiple forms, for example a date can appear as last Monday. We can never guarantee a completely exhaustive corpus. However, current QA systems have no reached performance levels where an exhaustive corpus is necessary to measure improvement. We use the results of our human annotations as an approximation to this ideal corpus. Lin only described the usefulness of this method as a replacement for the current black-box system evaluation but did not consider the additional use of this method for component analysis of QA systems.

Component Specific Evaluation Several groups have realised the importance of evaluating individual components independently of the whole system and have attempted to do so for specific components, for example, answer extraction (Light et al., 2001). Particular focus has been placed on the document retrieval component. For example, substituting different algorithms then comparing overall performance (Tellex et al., 2003). Also, human assessment of QA document retrieval has formed an additional part of the TREC QA Track (Voorhees, 2003) and the results are then used for automated evaluation (Monz, 2003). These individual analyses are useful, yet their application is limited to specific components and implementations. Our approach is not component specific.

Whole System Component Analysis There are two approaches to whole system component analysis. The first approach is assessing each component's usefulness or contribution to overall system performance. This can be done by ablation experiments (Brill et al., 2002), where each component is replaced by a baseline component which performs the same function in a minimal way.

The second approach is to look at cases where the system does not perform correctly, and identify the components which are causing these failures. Moldovan et al. (2003) manually traced each incorrectly answered question and decided which component was the cause. Each component was given a score in terms of the percentage of failures that it caused. This was used to determine which components of the system should be focused on to improve overall system performance. While useful, this analysis is time consuming and it is difficult to assign errors to specific components.

3 Collecting the Corpus

As the basis for the initial data collection we used the AQUAINT-1 document collection, consisting of approximately 1 million newswire articles, and a set of questions taken from the 1999 and 2000 TREC QA tracks. Also provided by NIST were expected answers and sets of regular expression answer keys intended for automatic marking.

Collecting an exhaustive list of correct answer instances is non trivial. We start with the assumption that humans are adept at finding specific information, in this case, the answers to questions. We organised a group of 20 volunteers to each spend 5 hours finding the answers to a set of TREC questions and annotating them. Reading through the entire document collection is infeasible. Thus, a web-based user interface was developed to assist the process, providing the annotators with a searching tool to first select a set of likely documents, then read them to find answers. All of the volunteer annotators were comfortable with keyword searching, primarily from the use of search engines such as Google.

Each annotator was allocated a question and given the expected answer and answers keys from TREC. They would then select some search keywords, and refine them until a manageable number of documents were returned. They would read through the documents and highlight any instances of the correct answer. Then they would create another search query and repeat the process until they were confident that they had found all of the answers.

The guidelines given to the annotators were as follows: The answer can exist in multiple forms, e.g. 1991, 25th March 1991, today. These should be tagged no matter what form or precision it is. Additionally, the whole answer should be tagged, if the document says 25th March 1991, then that whole phrase should be tagged. The annotators were also instructed that a correct answer must be supported by the sentence in which it is found, such that if a person read that sentence they would know, for certain, the answer to the question. This differs from the TREC evaluation process, which uses the entire document as support. The choice to allow only sentences both simplifies the annotation process and provides a closer correlation with how current QA systems select answers.

The questions were chosen based on the number of answers present in the document collection. This is impossible to determine without searching for, and verifying each answer. Therefore, the volunteer annotators also selected which questions would be included in the final set. Questions with fewer answer instances are most interesting for evaluation because systems are more likely to answer them incorrectly. For some questions, the number of correct answers in the document collection is overwhelmingly large and would be impossible for annotators to find all of them within a reasonable period. For these reasons, the annotators were instructed to discard questions when it became obvious that there would be more than 100 answer instances.

4 Behaviour of the Annotators

The use of human annotators was based on the assumption that humans, particularly those used to searching the internet for information, would be good at finding answers to questions. Looking at the behaviour of our volunteers, this is not the case. Humans are good at judging relevance but not at retrieval with high recall.

We analyse the shortcomings of the pilot corpus and discuss the necessary steps to enable efficient and reliable annotations of a more complete corpus.

4.1 Search Stopping Criteria

The annotators were specifically asked to find all instances of the answers for each question, however

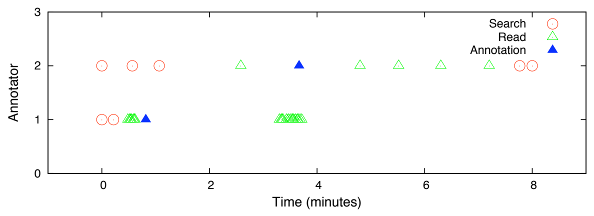


Figure 2: Time lines for What movie did Madilyn Kahn star in with Gene Wilder?

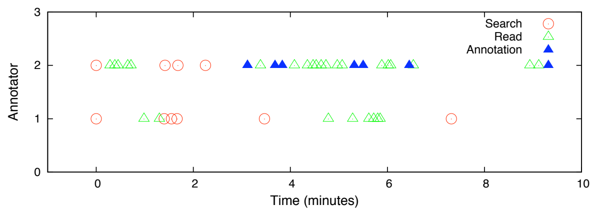


Figure 3: Timelines for Where is Tufts University?

there is no definite way to know when there are no more answers to be found. We found that the annotators used three different stopping criteria. (1) They had read all the documents returned from their searches and they believed that the search was general enough; (2) Further searches returned no new results; (3) After initially finding answers, if they read several documents with no answers.

We have plotted timelines showing when annotators searched for, read or annotated documents. Figure 2 shows two different stopping criteria. Annotator 1 stopped when there were no more documents in the search results and further searches yielded no results. Annotator 2 used a more general set of search terms and stopped annotating when they had read several documents without finding any answers. Both annotators found the same single answer.

This was not always the case. Figure 3 shows the behaviour of the two annotators assigned the question Where is Tufts University? Annotator 1, after several attempts at searching, decided that there were no answers for this question. Annotator 2 used more sophisticated queries, eventually finding several answers. From this variation in query generation ability, we know that not all of the answers have been found for each question.

Selecting the correct stopping criteria is not a trivial task and none of the criteria used by the annotators is sufficient in all circumstances.

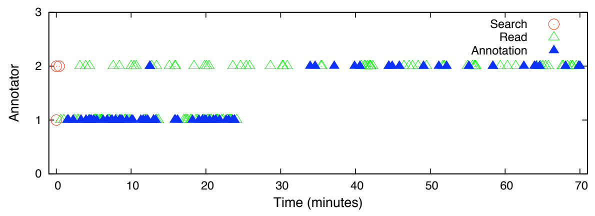


Figure 4: Time lines for Where is Venezuela?

4.2 Efficiency and Accuracy

The efficiency of the annotators related primarily to their choice of search terms. For example in the question Where is Venezuela?, shown in Figure 4. One annotator, not only was slower in reading and annotating documents, but also had fewer annotations within those documents because of the search terms used, they were less specific. The more efficient annotator searched for Venezuela AND 'south america' whereas the other annotator used keywords instead of a string literal and consequently had a higher proportion of irrelevant documents.

4.3 Search Terms

From a detailed analysis of the terms that the annotators used to retrieve documents we can both assess whether the annotators were thorough in finding answers exhaustively, and determine the annotators techniques for generating search queries.

For each question, the annotators were given the question, the expected answer and the set of answer keys. As shown in Table 1, the search terms used are usually generated from a combination of keywords from the question and from the expected answer. Unfamiliarity with regular expressions made using answer keys hard for the annotators.

The results in Table 1 were calculated based on simple case-insensitive string matching. Since the words were not stemmed, the search terms for the question, What did brontosauruses eat? had little overlap with the question because the annotators used the search term brontosaurus in combination with other words such as eat and food. The overall results suggest that the annotators used very few search terms which were not directly drawn from the question. In this example, the only word which is not a variation from the question is food.

Providing the answers for each question to the an-

Question	Overlap	Exp	Keys	Quest	None
A corgi is a kind of what?	**100**	50	50	50	0
What is sake?	50	**68**	37	25	0
What was the ball game of ancient Mayans called?	16	14	**100**	42	0
What's the average salary of a professional baseball player?	75	0	0	**100**	0
What did brontosauruses eat?	33	20	0	20	**80**
Averages	42	22	19	60	16

Table 1: The overlap of annotators search terms and the percentage source of the search terms, from the expected answer, answer keys and the question. The sample questions were chosen as the maximum of each column.

Text with annotation	#annotes	Correct
For instance , a 1993 fire that killed 188 workers at a toy factory in Thailand was similar to a **1911** New York clothing factory fire which previously held the record for the highest number of fatalities with 146 dead .	1	Incorrect
Some of the nation 's most stringent workplace-safety laws were prompted by the **1911** Triangle Shirtwaist Co. fire in New York City .	20	Correct
The decline has been particularly pronounced in high-risk industries such as mining , where the average death rate was 329 per 100,000 from **1911** to 1915 , compared with 25 per 100,000 in 1996-97 , the CDC said .	2	Incorrect
Here are some highlights : TRIANGLE FACTORY FIRE : On **March 25 , 1911** , a blaze at the Triangle Shirtwaist Co. in Manhattan 's garment district claimed 146 lives .	14	Correct
Here are some highlights : TRIANGLE FACTORY FIRE : On March 25 , **1911** , a blaze at the Triangle Shirtwaist Co. in Manhattan 's garment district claimed 146 lives .	4	

Table 2: A sample of the answer locations for the common question When was the Triangle Shirtwaist fire?

notators has further ramifications. In many cases, this lead to the annotators restricting their searches to only answers of a particular form. For example for the question When was Microsoft established? both annotators made the same search. Both annotators searched for Microsoft AND established AND 1975. Despite this high agreement, it is possible that further answers were completely disregarded, for example there could be answers of the form Microsoft was established 25 years ago today.

A total of 73% of the searches generated by the annotators contained either the expected answer or answer key in some form. The risk of making the queries answer specific is that answers in other formats will be missed. If the expected answer and answer keys were not provided, the annotators would be more likely to search for answers of any form. However, this has the disadvantage that the annotators would be less efficient in finding the answers.

The overall behaviour of the annotators focuses on speed and efficiency rather than accuracy or thoroughness, as the annotations were done within a short amount of time, with the goal of getting as much data as possible. With more time allowed for the annotations, more detailed guidelines and training, a far more exhaustive set of answer instances could be found for the question set.

4.4 Reliability of Manual Annotations

In order to study how accurately and exhaustively the annotators were finding the answers, all annotators were required to answer a common question: When was the Triangle Shirtwaist fire? Some results from this common question are shown in Table 2. These results show that not all of the answers which were marked were supported by the text surrounding them. Some annotators did not always follow the instruction that as much of the answer should be annotated as possible. However, these annotations are still considered correct because they overlap. With overlapping annotations the longest answer is used in the final answer set. Upon manual inspection, the majority of annotations for this question were correct, the highest agreement for an incorrect (unsupported) answer was 9, and the lowest agreement for a correct answer was 16 out of the 20 annotators. This is a significant gap and indicates that when there are enough annotators using only the answers which the majority of annotators agree upon is reliable both in terms of coverage and accuracy.

In addition to the common question, half of the set of questions were annotated by at least two annotators. A sample of questions and the overall annotation agreement results are shown in Table 3. The question Who was President Cleveland's wife? is an example of where both annotators failed to find all of the answers, both annotators found a different correct answer to the question. It is likely that there are more answers which were not found by either annotator. There were also clear mistakes in the dataset such as the question What is a nematode?, the two annotators assigned this question had annotated almost exactly the same sentences, however one of the annotators had mistakenly tagged the word nematode in each sentence and not the actual answer to the question.

In using this annotated answer set, there are two options, the first is to only use annotations which both annotators have agreed upon, this will significantly reduce the erroneous answers, however it would also reduce the coverage of the answer set. When using the answer set to analyse a QA system, the system would appear to be performing worse than it actually is. The other option is to include all answers regardless of whether multiple annotators agreed on them. This would result in the system appearing worse than it actually is when it does not retain the wrong answer, however it will appear to be performing better than it actually is if it retains the wrong answers. We chose to include all of the answers to maximise coverage, as this it is coverage which is one of the faults of the current standard evaluation system. Ideally, there would be more annotators for each question, and thus the inclusion of each answer location could be decided with relative confidence by a majority vote, as shown in result of the common question described above.

5 Using Answer Attenuation

When constructing or modifying a QA system, there is a trade-off between speed, memory and how much of the document collection is retained at each stage. For example, if at the document selection stage, all of the documents are selected, this guarantees that all answers in the document collection are still within the search space, however the following stages will take much more time to process this larger search space. Alternatively only one document or passage could be selected at each stage. This would result in very fast processing in the later stages however most of the answers will be lost and assuming each stage involves some probability of error, it is possible that all correct answers are lost resulting in the system returning the wrong answer or no answer. Consequently a QA system designer must balance these two concerns: efficiency and recall. This is where our answer attenuation measure is critical.

Using the answer set we collected, it is possible to measure the proportion of this answer set which is still in the QA system's search space after each stage of processing each question. What defines a stage of processing, is any work done by the system which narrows the search space for answers, whether this is intentional or not. For example, document retrieval is intentionally narrowing the search to a specific set of documents, whereas a parsing step removes sentences if they fail to parse, an effect which is unintentional. Consequently, answer attenuation serves the dual purpose of error detection as well as an analysis tool to fine tune the QA system to maximise performance.

6 Results

We compared answer attenuation to the TREC answer keys in two ways. Firstly, we compared all of the annotated answers to check whether they would have been marked as correct by the TREC answer keys. Secondly, we calculated the answer attenuation for all six stages of the Pronto QA system. The evaluation of the final answer, we compared with the TREC answer key evaluation. Our final evaluation of answer attenuation is to show that it is useful in detecting and repairing flaws in QA systems.

6.1 Comparing Annotations with Answer Keys

The final answers produced by Pronto were marked by both the TREC answer keys and whether the final

Question	Annotations	Answers	Aggreement
Who was President Cleveland's wife?	2	2	0%
When was Microsoft established?	2	1	100%
What is a nematode?	16	16	0%
Where does chocolate come from?	31	23	34%
Where is Trinidad?	22	21	4%
Who was the 33rd president of the United States?	10	5	100%
What province is Edmonton located in?	34	28	21%
Average for all questions	17.0	13.9	31.2%

Table 3: A sample of questions and the agreement when a question was answered by two annotators

answer location was one of those in the answer set. A difference between these two scores must have one of three possible causes: (1) there is a correct answer in the document collection which the annotators did not find; (2) the answer key marks the answer as correct when it is not supported by the document; (3) the answer returned was annotated incorrectly by the annotators. In all of the experiments run, this third case never occurred, despite Pronto answering most of the questions wrong, and the existence of several incorrect answers in the answer set. The question What is the busiest air travel season? is an example of the TREC answer key marking as correct when the document does not actually answer the question. The source document refers to a specific summer in which there were many air traffic delays, not that the season of summer is the busiest time to travel or even busy at all. Similarly for the question Where is Tufts University?, the answer Boston is correct according to the TREC answer keys but the document from which it was taken does not support it.

That document was read by both annotators who were assigned that question yet was not annotated, indicating their belief that it did not answer the question. In every case where the returned answer was not one of the annotations and the TREC answer key marked it as correct, the document was found by a human assessment to not support the answer. This drawback in the answer key method is clearly shown by marking all of the human annotations according to the TREC answer keys, the results of this comparison is shown in Table 4. Only 42% of the answers found by the annotators would be considered correct according to an evaluation based solely on the answer key. Additionally for some of the questions, none of the answers marked by the annotators would have been marked as correct by the TREC answer key.

7 Answer Attenuation in Pronto

The Pronto QA system has 6 discrete components which actively reduce the search space. The document retrieval component selects the document identifiers from an index based on search terms, these documents are extracted into a single file which is parsed with the wide-coverage CCG Parser (Clark and Curran, 2004). This syntactic analysis is then used by a component called Boxer to generate a semantic analysis of the text (Bos, 2005). The document retrieval and extraction components form the document selection stage according to the generic QA system structure. Similarly, the CCG and Boxer stages form the document analysis stage. The matching component performs answer extraction, and 'select' refers to the answer ranking stage. Pronto is currently in a state of development and is consequently not performing to the standard expected of a state of the art system.

Component	Q	%L	Run1	%L	Run2	%L
Total Ans.	10	0	872	0	872	0
DocRet	4	60	493	43	493	43
Extract	4	0	418	15	493	0
CCG	4	0	345	17	415	15
Boxer	4	0	345	0	415	0
Matcher	1	75	30	91	27	93
Select	1	0	5	83	4	85
Answer	1	0	3	40	3	25

Table 5: The answer attenuation for Pronto. Showing the answers remaining in the search space and the % loss for an example question (Q) and two runs.

We measured the number of correct answers in the search space at the beginning and end of each

Statistic	Min	Max	Total	Avg. per Question
Number of annotations	0	339	1532	14.7
Number of annotations which matched answer keys	0	336	833	8.0
Percent of annotations correct according to answer keys	0%	100%	54.4%	41.6%
Number of answers annotated	0	69	1058	10.2
Number of answers which matched answer keys	0	44	446	4.3
Percent correct according to answer keys	0%	100%	42.2%	41.1%

Table 4: For each question, the number of answers the annotators found and the percentage of these which would be judged correct by the answer keys

stage. An example question, What is the wingspan of a condor? is shown in Table 5, there were a total of 10 answers in the annotated answer set for this question. Only 4 answers were found in the document retrieval stage, and 3 of those answers were lost in the matching stage. The system returned the remaining correct answer as its final result. The first run shows the initial results aggregated over all question. We expected the document extraction phase to have zero loss. This stage only copies the documents from the collection into a single file and is not intended to reduce the search space. This was traced to a problem in the way Pronto handles results form multiple searches and the problem repaired for the second run. This repair improved the individual component score, yet did not improve the overall system score, hence by black box evaluation the repair would not be detectable.

8 The Ultimate Answer Set

Our results indicate that with a larger group of annotators, a more complete set of answers could be found. We propose the creation of a collaborative dataset, emulating the success of open projects such as Wikipedia. Using a fixed document collection and question set, an initial set of answer locations is used as an evaluation set alongside the TREC answer keys and manual evaluations. Whenever a new answer is found, it is added to the dataset if a threshold number of humans agree on the answer. Similarly, if an answer is included in the collection which is agreed to be incorrect or not have enough support from the source document, it is removed. Over time the answer set is refined to be both exhaustive and accurate.

9 Conclusion

The answer attenuation measure provides both a useful analysis of QA system components and has the potential to also be a reliable whole system performance evaluation. This depends on the creation of a sufficiently exhaustive corpus of answer locations. We have created a pilot corpus of answer locations, and by studying the behaviour of the annotators we conclude that it is not yet reliable enough to replace current methods of overall evaluation. What was lacking in QA evaluation was an automated method for conducting an evaluation across all system components.

The next step is the creation of a larger high quality answer location corpus. This would involve more detailed guidelines for the annotators, more annotators assigned to each question and more time allowed. The resulting corpus could then be improved collaboratively over time as it is used to evaluate QA systems.

The creation of our pilot corpus has shown that this goal is feasible. We have used this corpus to analyse answer attenuation in Pronto, and have shown that it can reveal flaws in existing question answering systems.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback, Johan Bos who helped with our installation of Pronto, and our 20 volunteer annotators for helping us test these ideas.

References

Ion Androutsopoulos. 1995. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1:29–81.

Johan Bos, James R. Curran, and Edoardo Guzzetti. 2007. The Pronto QA System at TREC 2007: Harvesting Hyponyms, Using Nominalisation Patterns, and Computing Answer Cardinality. In *TREC 2007*.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*, pages 42–53.

Eric J. Breck, John D. Burger, Lisa Ferro, Lynette Hirschman, David House, Marc Light, and Inderjeet Mani. 2000. How to Evaluate your Question Answering System Every Day and Still Get Real Work Done. In *LREC-2000*, pages 1495–1500.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the askmsr question-answering system. In *EMNLP '02*, pages 257–264, Morristown, NJ, USA. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*.

L. Hirschman and R. Gaizauskas. 2001. Natural language question answering: the view from here. *Nat. Lang. Eng.*, 7(4):275–300.

Marc Light, Gideon S Mann, Ellen Riloff, and Eric Breck. 2001. Analyses for elucidating current question answering technology. *Natural Language engineering*, 7(4):325–342.

Jimmy Lin. 2005. Evaluation of resources for question answering evaluation. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 392–399, New York, NY, USA. ACM.

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21(2):133–154, April.

Christof Monz, 2003. *Document Retrieval in the Context of Question Answering*, chapter 8, page 546. Springer Berlin / Heidelberg.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03*, pages 41–47, NY, USA. ACM.

Ellen M. Voorhees. 2003. Overview of the trec 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*. National Institute of Standards and Technology.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 11-18

Using multiple sources of agreement information for sentiment classification of political transcripts

Clint Burfoot
Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
c.burfoot@pgrad.unimelb.edu.au

Abstract

Sentiment classifiers attempt to determine whether a document expresses a generally positive or negative sentiment about its topic. Previous work has shown that overall performance can be improved by combining per-document classifications with information about agreement between documents. This paper explores approaches to sentiment classification of U.S Congressional floor debate transcripts that use a model for incorporating multiple sources of information about agreements between speakers. An empirical evaluation demonstrates accuracy improvements over previously published results.

1 Introduction

Thomas et al. (2006) investigate whether one can determine from the transcripts of U.S. Congressional floor debates whether speeches support or oppose a piece of proposed legislation. The authors exploit the fact that speeches occur as part of a discussion in order to incorporate information about their inter-relationships. Using this information provides substantial improvements over analysing speeches in isolation. The authors adopt a model, based on graph mincuts, that optimally balances classifications of speeches performed in isolation with information about whether or not pairs of speeches are likely to agree.

The work reported here mirrors Thomas et al. in using the same corpus and attempting the same binary sentiment classification task. It refines the the concept of agreement, comparing alternate techniques for detecting agreement in debates, and showing how these techniques can improve performance in the task of classifying speeches.

Two types of agreement information are introduced. The first, *party* agreement, takes advantage of the fact that speakers tend on average to vote with the majority of their party. A party-classifier is trained to detect whether a speaker is most likely a Republican or a Democrat. Links between pairs of speeches assigned to the same party encourage the overall classifier to give these speeches the same label. The second type of agreement is based on the intuition that speakers who agree often use similar words to describe their position. This *similarity* agreement is derived by comparing the context windows that surround references to the bill being considered in the debate.

An experimental evaluation shows that incremental accuracy improvements can be gained by making use of this agreement information. A range of technical limitations are discussed and ideas considered for future work that may make even better use of agreement information to improve sentiment classification.

2 Related Work

For a detailed survey of the field of sentiment analysis see (Pang and Lee, 2008).

The document-level sentiment-polarity classification task began as a “thumbs up or thumbs down” experiment. Pang et al. (2002) apply machine learning methods to predicting the overall sentiment of movie reviews. Turney uses PMI-IR to perform sentiment classification of reviews of banks, movies, cars and travel destinations (Turney, 2002; Turney, 2001).

Much work has focussed on the problem of separating language that contributes to an understanding of the sentiment of a document from language that is merely noise. Pang and Lee (2004) describe a sentence *subjectivity detector* that is trained on sets of labelled subjective and objective sentences. It embodies the intuition that important sentences will tend to cluster together by encouraging sentences to be classified as subjective or objective depending upon the classifications of their neighbours. Mullen and Collier (2004) isolate adjectives that occur in the same sentence as a reference to the topic of the document. Xia et al. (2008) use a semantic lexicon to extract sentiment words and surrounding words that serve to negate or modify the sentiment.

Work on document-level sentiment-polarity classification has built on attempts at determining *semantic orientation* of adjectives, i.e. whether an adjective indicates positive or negative sentiment. Kamps et al. (2004) grade adjectives on the bipolar adjective scales good/bad, active/passive and strong/weak using WordNet synonymy relationships. They allocate an adjective a score on each scale by finding the relative number of synonymy links that have to be traversed to get to the two pole adjectives. Hatzivassiloglou and McKeown (1997) extract sets of positive and negative adjectives from a large corpus using the insight that conjoined adjectives are generally of the same or different *semantic orientation* depending open the particular conjunction used. The results are specific to the domain of the corpus, which is helpful when adjectives have domain-specific orientations.

Some more recent sentiment classification papers have focussed on combining training data from multiple domains (Andreevskaia and Bergler, 2008; Li and Zong, 2008).

A number of studies have focussed on identifying agreement between speakers in the context of multi-party conversations. Galley et al. (2004), for example, describe a statistical model for identifying adjacency pairs and deciding if an utterance indicates agreement or disagreement with its pair. They classify using lexical (including *semantic orientation*), durational and structural features.

3 Corpus

Thomas et al. created the ConVote corpus using GovTrack[1], an independent website that collects publicly available data on the legislative and fund-raising activities of U.S. congresspeople. The HTML versions of GovTrack's floor-debate transcripts for 2005 were downloaded, cleaned, tokenised, and broken into debates and constituent speeches for use in the corpus.

Each speech is labelled with an identifier for its speaker and his recorded vote for the corresponding debate. Debates in which the losing side obtained less than 20% of the vote are omitted from the corpus on the grounds that they have less interesting discourse structure.

The debates are randomly allocated to training, development and test sets.

The text of each speech is parsed for references to other speakers (e.g. "I was pleased to work with the committee on the judiciary, and especially the gentleman from Virginia my good friend, to support the legislation on the floor today"), which are automatically tagged with the identity of the speaker being referred to.

An alternate version of the corpus is provided for use in classifying speeches in isolation. Agreement tags are not included. Short speeches that refer to yielding time (e.g. "Madame Speaker, I am pleased to yield 5 minutes to the gentleman from Massachusetts?") are removed on the grounds that they are purely procedural and too insubstantial for classification. Speeches containing the word "amendment" are removed because they tend to contain opinions on amendments rather than the bill being discussed.

4 Method

4.1 Classifying speeches using per-speech and inter-speech information

This work follows the method of Thomas et al. for incorporating information about inter-speech relationships into an overall classification. They draw on Blum and Chawla (2001), who describe how graph mincuts can balance per-element and pairwise information.

[1]http://govtrack.us

	total	train	test	development
speeches	3857	2740	860	257
debates	53	38	10	5
average number of speeches per debate	72.8	72.1	86.0	51.4
average number of speakers per debate	32.1	30.9	41.1	22.6

Table 1: Corpus statistics. Taken from Thomas et al. (2006)

Following Thomas et al., let $s_1, s_2, \ldots, s_n$ be the speeches in a given debate and let $\mathcal{Y}$ and $\mathcal{N}$ stand for the "supporting" and "opposing" classes, respectively. Assume a non-negative function $\mathit{ind}(s, C)$ indicating a degree of preference for assigning speech s to class C. Also, assume some pairs of speeches have a link with non-negative strength, where str(ℓ) for a link ℓ indicates a degree of preference for the linked speeches to be assigned to the same class. The class allocation $c(s_1), c(s_2), \ldots, c(s_n)$ is assigned a cost

$$\begin{aligned} c &= \sum_s \mathit{ind}(s, \bar{c}(s)) \\ &+ \sum_{s,s':\ c(s)\neq c(s')} \ \sum_{\ell \text{ between } s,s'} \mathrm{str}(\ell) \end{aligned} \qquad (1)$$

where $\bar{c}(s)$ is the complement class to $c(s)$. A *minimum-cost* assignment then represents an optimum way to balance a tendency for speeches to be classified according to their individual characteristics with a tendency for certain speech pairs to be classified the same way. The optimisation problem can be solved efficiently using standard methods for finding minimum cuts in graphs.

4.2 Classifying speeches in isolation

Whole-of-speech classification: Pang et al (2002) show that standard machine-learning methods can be used to classify documents by overall sentiment, with best results coming from support vector machines (SVM) with unigram presence features for all words. Punctuation characters are retained and treated as separate tokens and no stemming is done. Their approach is adopted to classify speeches in isolation using the popular SVM^{light} [2] classifier with default parameters (Joachims, 1999).

[2]http://svmlight.joachims.org/

Following Thomas et al. speeches are allocated an *ind* value based on the signed distance $d(s)$ to the SVM decision plane:

$$\mathit{ind}(s, \mathcal{Y}) \overset{\text{def}}{=} \begin{cases} 1 & d(s) > 2\sigma_s; \\ \left(1 + \frac{d(s)}{2\sigma_s}\right)/2 & |d(s)| \leq 2\sigma_s; \\ 0 & d(s) < -2\sigma_s \end{cases}$$

where σ_s is the standard deviation of $d(s)$ over all of the speeches s in the debate and $\mathit{ind}(s, \mathcal{N}) = 1 - \mathit{ind}(s, \mathcal{Y})$.

4.3 Classifying agreements between speeches

Same-speaker agreements: Speakers in congressional floor-debates will often contribute more than one speech. As Thomas et al. note, one can imagine that if a political debate is serving its purpose a speaker might change his mind during its course, so that one of his later speeches contradicts an earlier one. Unfortunately, this scenario has to be ruled of consideration since our corpus labels speeches based on the speaker's final vote. To represent this simplification, each of the speeches by a given speaker is linked to another with a link ℓ of infinite strength. This guarantees that they will receive the same final classification [3].

Reference agreements: Speakers in congressional floor-debates sometimes refer to each other by name. These references are labelled in the corpus. Thomas et al. build an agreement classifier to take advantage of the intuition that the words a speaker uses when referring to another speaker will give a clue as to whether the two agree. They use an SVM classifier trained on the tokens immediately surrounding[4] the reference. Following Thomas et al.

[3]Detecting the point at which speakers change their minds could make an interesting area for further research.

[4]The authors find good development set performance using the window starting 30 tokens before the reference and ending 20 tokens after it.

let $d(r)$ denote the distance from the vector representing the reference r to the SVM decision plane and let σ_r be the standard deviation of $d(r)$ over all references in the debate. Define the strength *str* of the link as:

$$\mathit{str}(r) \overset{\text{def}}{=} \begin{cases} 0 & d(r) < \theta_{\text{agr}}; \\ \alpha_{\text{r}} \cdot d(r)/4\sigma_r & \theta_{\text{agr}} \leq d(r) \leq 4\sigma_r; \\ \alpha_{\text{r}} & d(r) > 4\sigma_r \end{cases}$$

where θ_{agr} is a threshold that increases the precision of the links by discarding references that are not classified with enough confidence and α_{r} represents the relative strength of reference links.

Same-party agreements: A brief examination of the ConVote corpus confirms the intuition that speakers tend to vote with their party. This is a form of agreement information. If we know that two parties "agree" on their choice of party affiliation, we can conclude they are more likely to vote the same way on any given bill. The method already described for whole-of-speech classification can be trivially extended for party detection by substituting party labels for vote labels.

It is reasonable to assume that by-name references to other speakers also give a hint about party affiliation. A reference classifier is trained with party labels, using the method described in the section above.

An overall party classification, $p(s)$, is derived for the whole-of-speech and reference agreement classifiers using the graph mincut method already described. Define the strength $\mathit{pstr}(s, s')$ of the agreement link as:

$$\mathit{pstr}(s, s') \overset{\text{def}}{=} \begin{cases} \alpha_{\text{p}} & p(s) = p(s'); \\ 0 & p(s) \neq p(s') \end{cases}$$

where α_{p} represents the relative strength of party links.

Similarity agreements: To measure the extent to which a pair of speakers use similar words to describe their positions, let $\mathit{sim}(s, s')$ be the similarity of two speeches in a debate determined with the standard information retrieval measure of cosine similarity with tf.idf term weighting (Manning et al., 2008). Define a link strength $\mathit{bstr}(s, s')$ as:

$$\mathit{bstr}(s, s') \overset{\text{def}}{=} \begin{cases} \mathit{sim}(s, s') \cdot \alpha_{\text{b}} & \mathit{sim}(s, s') > 4\sigma_{\text{b}}; \\ 0 & \mathit{sim}(s, s') \leq 4\sigma_{\text{b}} \end{cases}$$

where σ_{b} is the standard deviation of $\mathit{sim}(s, s')$ over all of the speeches in the debate and α_{b} represents the relative strength of similarity links. The use of the threshold based on standard deviation serves to limit the links to the most strongly similar pairs without introducing another free parameter.

To reduce noise, the input to the similarity algorithm is limited to the set of tokens that appear within a fixed window of the tokens "bill" or "h.r."[5]. These two tokens tend to indicate that the speaker is commenting directly on the bill that is the topic of the debate.

Overall classification: To incorporate these two new measures of agreement into the overall classification framework, assign a new value for *c* by modifying equation (1) as

$$\begin{aligned} c &= \sum_{s} \text{ind}(s, \bar{c}(s)) \qquad (2) \\ &+ \sum_{s,s'\colon c(s)\neq c(s')} \left\{ \begin{array}{rl} & \mathit{pstr}(s, s') \\ + & \mathit{bstr}(s, s') \\ + & \sum_{\ell \text{ between } s,s'} \text{str}(\ell) \end{array} \right\} \end{aligned}$$

4.4 A note on relevance

Impressionistically, a significant proportion of speeches are not clearly relevant to the bill. Even with the "amendment" and "yield" speeches removed, there are a variety of procedural utterances and instances where speakers are evidently dealing with some unrelated bill or motion. These spurious speeches are problematic as they skew the final accuracy figures and may reduce classifier accuracy by introducing noise. An early iteration of this work used official summaries of the bills to build a relevance metric. Speeches that had high tf.idf similarity with the bill summary were considered more relevant. This figure was used to test three hypotheses.

1. Speeches that have a higher relevance will be more accurately classified by the whole-of-speech classifier.

[5]Good development set performance is obtained using the window starting 15 tokens before the reference and ending 15 tokens after it.

Similarity agreement classifier (“similarity⇒agreement?”)	Devel. set	Test set
majority baseline	49.02	49.02
classifier	61.09	59.94

Table 2: Similarity agreement accuracy, in percent.

2. Reference agreements between pairs of speeches that have higher relevance will be classified more accurately.

3. Similarity agreements between pairs of speeches that have higher relevance will be more accurate.

Early experiments did not support these hypotheses, so the approach was abandoned. Nevertheless a more sophisticated measure of relevance may eventually prove to be a key to improved accuracy.

5 Evaluation

This section presents experiments intended to evaluate the performance of the classifier against baselines and benchmarks. Ten-fold cross validation is used for all but one experiment, with 8 parts of the data designated for training, 1 for testing and 1 for development. The development set is used for tuning free parameters. The tuning process consists of repeatedly running the experiment with different values for the free parameters α_r, α_p and α_b. The combination of values that gives the best result is then used for final evaluation against the test set.

The last experiment is a comparison with the results from Thomas et al.

5.1 Similarity agreements

The baseline for similarity agreement classification is the percentage of possible speech pairs that agree. This is approximately half. Table 2 shows that the classifier predicts agreement with about 60% accuracy.

5.2 Reference classification

The baseline for reference classification is the percentage of by-name references that correspond with agreement across the whole corpus. The relatively high figure is evidence that speakers tend to refer to the names of others with whom they agree. The value for θ_{agr} is not optimised. Just two values were tried: 0 and μ, the average decision-place distance across all non-negative scores. As shown in Tables 3 and 4, the use of a non-zero cutoff introduces a precision-recall tradeoff.

Agreement classifier (“reference⇒agreement?”)	Devel. set	Test set
majority baseline	81.48	80.23
$\theta_{agr} = 0$	80.39	80.80
$\theta_{agr} = \mu$	89.59	89.48

Table 3: Agreement-classifier accuracy, in percent. References that do not meet the threshold are not counted.

Same-party classifier (“reference⇒same-party?”)	Devel. set	Test set
majority baseline	77.16	79.68
$\theta_{agr} = 0$	81.31	76.23
$\theta_{agr} = \mu$	81.44	78.74

Table 4: Same-party-classifier accuracy, in percent. References that do not meet the threshold are not counted.

An early version of this experiment attempted to infer agreements from disagreements. Two speakers who disagree with a third speaker must, by definition, agree with each other, since speakers can only vote to support or oppose. Two factors limited the usefulness of this approach. First, less than 20% of references correspond with disagreement. Speakers seem to prefer referring by name to others with whom they agree. Second, the agreement classifier did not do a reliable job of detecting these.

5.3 Party classification

An audit of the corpus shows that, averaged across all debates, 92% of votes concur with the party majority. This should mean that the 85% accurate labels obtained by the party classifier, shown in Table 5 should make prediction of votes significantly easier.

An alternative to party classification would have been to take the party labels as input to the vote classifier. After all, it is reasonable to assume that any real-world application of sentiment classification of formal political debate could rely on knowledge of the party affiliation of the speakers. Two factors make it more interesting to limit the use of prior

Republican/Democrat classifier ("speech⇒Republican?")	Devel. set	Test set
majority baseline	51.08	51.08
SVM [speech]	68.97	69.35
SVM + same-speaker-links ...		
+ agreement links; $\theta_{\text{agr}} = 0$	81.31	76.23
SVM + same-speaker-links ...		
+ agreement links; $\theta_{\text{agr}} = \mu$	85.22	84.26

Table 5: Republican/Democrat-classifier accuracy, in percent.

Support/oppose classifier ("speech⇒support?")	Devel. set	Test set
majority baseline	53.85	53.85
SVM + same-speaker-links ...		
+ agreement links, $\theta_{\text{agr}} = \mu$	81.77	79.67
+ agreement links, $\theta_{\text{agr}} = \mu$		
+ party links	85.07	80.50
+ agreement links, $\theta_{\text{agr}} = \mu$		
+ similarity links	81.77	79.67

Table 6: Support/oppose classifier accuracy, in percent.

knowledge for the purposes of this paper. First, there are cases where party affiliation will not be available. For example, it is helpful to know when independent speakers in a debate are using language that is closer to one party or the other, since their vote on that debate will probably tend accordingly. Second, this approach better demonstrates the validity of the classification model for use with imperfect domain knowledge. Such techniques are more likely to be applicable in informal domains such as analysis of political text in the blogosphere.

5.4 Overall classification

The benchmark for evaluating the utility of party and similarity agreements is the score for in-isolation classification combined with same speaker links and reference agreements. This is represented in Table 6 as "SVM + same-speaker-links + agreement links, $\theta_{\text{agr}} = \mu$". Adding in party links improves accuracy by about 1% on the test set and 3% on the development set.

Adding similarity links does not improve overall accuracy. It is somewhat surprising that party links do better than similarity links. Similarity links have the advantage of existing in variable strengths, depending upon the degree of similarity of the two speeches. It may be that the links are simply not reliable enough. It seems likely that the three relatively simple methods used in this work for detecting agreement could be improved with further research, with a corresponding improvement to overall classification accuracy.

5.5 Comparison with previous results

Thomas et al. obtained a best result of 70.81% with $\theta_{\text{agr}} = 0$ and no cross-validation. The equivalent best result produced for this experiment using the same algorithm was 71.16%, a difference probably due to some minor divergence in implementation. The addition of party links gives no accuracy increase. Adding similarity links gives an accuracy increase of about 0.5% on both the development and test sets.

As shown in Table 7, the development set results for the Thomas et al. experiment are much better than those obtained using cross-validation. The test set results are much worse. This is surprising. It appears that the Thomas et al. split is unlucky in the sense that differences between the development and test sets cause unhelpful tuning. The use of cross validation helps to even out results, but the arbitrary split into sets still represents a methodological problem. By using 8 sets of debates for training, 1 for testing and 1 for tuning, we are open to luck in the ordering of sets because the difference between good and poor results may come down to how well the development set that is chosen for tuning in each fold happens to suit the test set. One solution would be to cross-validate for every possible combination of test, development and training sets.

5.6 Choice of evaluation metric

Since our choice of ground truth precludes the speeches by a speaker in a single debate from having different labels, the decision to evaluate in terms of percentage of speeches correctly classified seems slightly questionable. The alternative would be to report the percentage of speakers whose votes are

Support/oppose classifier (“speech⇒support?”)	Devel. set	Test set
majority baseline	54.09	58.37
SVM + same-speaker-links ...		
+ agreement links, $\theta_{\text{agr}} = \mu$	89.11	71.16
+ agreement links, $\theta_{\text{agr}} = \mu$ + party links	89.11	71.16
+ agreement links, $\theta_{\text{agr}} = \mu$ + similarity links	89.88	71.74

Table 7: Support/oppose classifier accuracy, in percent, using the training, development, test debate split from Thomas et al. without cross-validation.

correctly predicted. This approach more correctly expresses the difficulty of the task in terms of the number of degrees of freedom available to the classifier, but fails to consider the greater importance of correctly classifying speakers who contribute more to the debate. This work has retained the established approach to allow comparison. More comprehensive future works might benefit from including both measures.

6 Conclusions and future work

This study has demonstrated a simple method for using multiple sources of agreement information to assist sentiment classification. The method exploits moderately reliable information about whether or not documents agree in order to improve overall classification. Accuracy suffers somewhat because of the need to tune link strengths on a set of development data. Future work should attempt to remove this limitation by developing a more principled approach to incorporating disparate information sources.

There is great scope for exploration of the concept of agreement in sentiment analysis in general. Being able to detect whether or not two documents agree is likely to be useful in areas beyond sentiment classification. For example, tools could assist researchers in understanding the nuances of contentious issues on the web by highlighting areas in which different sites or pages agree and disagree. eRulemaking tools that cross-link and group public submissions about proposed legislation could benefit from being able to match like opinions. Matching could be on the basis of overall opinion or a breakdown of the issue into separate aspects. Sentiment summarisation tools could be aided by the ability to group content from separate documents into sections that have been judged to have equivalent sentiment.

Document-level sentiment classification techniques are limited by their inability to reliably ascribe expressions of sentiment. Strongly positive or negative expressions may relate to an aspect of the topic that is linked to overall sentiment in an unpredictable way. For example, a speaker offers the following to a debate on establishing a committee to investigate the preparation for, and response to, Hurricane Katrina: “Mr. Speaker, the human suffering and physical damage wrought by Hurricane Katrina is heart-wrenching and overwhelming. We all know that very well. Lives have been lost and uprooted. Families are separated without homes and without jobs.” This extract is full of strongly negative words. Nevertheless, it comes from a speech given in support of the bill. It is unlikely that a bag-of-words polarity classifier will be able to separate the negative sentiment expressed about Hurricane Katrina from any sentiment that is expressed about the bill itself.

As another example, the following quotes are from two contributors to a debate on protected species legislation. “The E.S.A has only helped 10 of 1,300 species listed under the law. Thirty-nine percent of the species are unknown. Twenty-one percent are declining, and they are declining, and 3 percent are extinct. This law has a 99 percent failure rate.” “Mr. Chairman, the endangered species act is a well-intentioned law that has failed in its implementation. Originally billed as a way to recover and rehabilitate endangered species, it has failed at that goal.” Both of these quotes use apparently negative language, yet they are given in support of the proposed legislation. The negative opinion being expressed is about legislation which is to be replaced by the bill under debate.

One way for a classifier to deal with this kind of material is to make the rhetorical connection between negative sentiment about an existing bill and support for its replacement. This is quite a challenge. An alternative is to make use of agreement links. If the classifier can successfully detect that the negative sentiment in both speeches relates to the "E.S.A." it can correctly determine that the two agree on at least part of the issue. This information can then be combined with a judgement from a per-document classifier that is trained to consider only parts of speeches that refer directly to the bill being discussed. A three-stage process might be: (i) Performing in-isolation classification based on a document extract that is deemed to express overall sentiment; (ii) Performing agreement classification on document extracts that are deemed to relate to distinct aspects of the debate; and (iii) Completing an overall classification based on these per-document and inter-document measures. Future work could focus on developing this approach.

References

Alina Andreevskaia and Sabine Bergler. 2008. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. In *Proceedings of ACL-08: HLT*, pages 290–298, Columbus, Ohio, June. Association for Computational Linguistics.

Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 669–676, Barcelona, Spain, July.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 174–181, Madrid, Spain, July. Association for Computational Linguistics.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, Cambridge, MA, USA.

J. Kamps, M. Marx, R. Mokken, and M. de Rijke. 2004. Using WordNet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, volume IV, pages 1115–1118.

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *Proceedings of ACL-08: HLT, Short Papers*, pages 257–260, Columbus, Ohio, June. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 412–418, Barcelona, Spain, July. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 271–278, Barcelona, Spain, July.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundation and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 Conference on Empirical methods in Natural Language Processing*, pages 79–86, Morristown, NJ, USA. Association for Computational Linguistics.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335, Sydney, Australia, July. Association for Computational Linguistics.

Peter D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *ECML '01: Proceedings of the 12th European Conference on Machine Learning*, pages 491–502, London, UK. Springer-Verlag.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Yunqing Xia, Linlin Wang, Kam-Fai Wong, and Mingxing Xu. 2008. Lyric-based song sentiment classification with sentiment vector space model. In *Proceedings of ACL-08: HLT, Short Papers*, pages 133–136, Columbus, Ohio, June. Association for Computational Linguistics.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 19-27

All-Topology, Semi-Abstract Syntactic Features for Text Categorisation

Ari Chanen
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
ari@it.usyd.edu.au

Jon Patrick
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
jonpat@it.usyd.edu.au

Abstract

Good performance on Text Classification (TC) tasks depends on effective and statistically significant features. Typically, the simple bag-of-words representation is widely used because unigram counts are more likely to be significant compared to more compound features. This research explores the idea that the major cause of poor performance of some complex features is sparsity. Syntactic features are usually complex being made up of both lexical and syntactic information. This paper introduces the use of a class of automatically extractable, syntactic features to the TC task. These features are based on subtrees of parse trees. As such, a large number of these features are generated. Our results suggest that generating a diverse set of these features may help in increasing performance. Partial abstraction of the features also seems to boost performance by counteracting sparsity. We will show that various subsets of our syntactic features do outperform the bag-of-words representation alone.

1 Introduction

One of the keys to obtaining good performance out of an automatic TC system is choosing appropriate types of features. Humans can often do better than automatic classification systems on difficult-to-categorise corpora, probably by using the human ability to extract the **meaning** of documents from the document words. Many researchers have tried to use features that are closer to a more semantic representation of documents. Unfortunately, many of these attempts do not succeed in doing better than bag-of-words e.g. (Moschitti and Basili, 2004). A major problem seems to be that the more complex a feature type is, the less frequently it occurs. A complex feature's occurance rate in a training set will often not be significant in an unseen test set. A simple example is how a purely bigram representation of documents is expected to do worse than a representation of the same documents using unigrams, as we will demonstrate below in the Experiments section.

Attempting to use syntactic features can be appealing because the syntax of document sentences holds clues into how an author encodes the semantics into the sequence of document words. Syntax can be thought of as the middle ground between the lexical and semantic levels of representation. An apt quote on this subject is "...the aim of syntactical representation is to constitute a bridgehead to semantics"(Schneider, 1998). The syntax of a document is usually represented in a hyper-linear representation such as trees. Syntactic trees are rich in information where the nodes encode information about sentence terms and the links encode information about the relationships between terms.

It is tempting to construct complex features by extracting them from parse trees. However, there is always the problem of feature sparsity. The right type of complex, syntactic feature may be closer to the semantic level and may seem to hold out the promise of improving performance on a TC task, yet more complex phrasal or syntactic features suffer from "inferior statistical qualities" (Lewis, 1992).

It is the aim of this research to try to use syntactic information in TC features in such a way that the problems of sparsity might be overcome. In this effort, two main lines of attack are used. First, we automatically generate a large number (possibly hundreds of thousands or more) of features out of dependency parse trees, in the hope that some of these many features will turn out to be highly discriminative TC features with a significant number of occurrences. Second, we use partial abstraction of these syntactic features to bring separate groups of features together, thus increasing the feature count and alleviating sparsity.

It should be noted that sparsity may be only one reason that complex features may not be as effective as simple features given a particular task. Other reasons might be that some types of complex features are simply noisy with respect to a learning task. Another possible reason is that some simple features subsume more complex features with respect to a task. These two reasons beyond sparsity will not be explored in this research.

The TC task that we do our experiments on is that of scam detection i.e. separating scam websites from non-scam websites in the financial domain. The dataset comes from the ScamSeek project (Patrick, 2006b).

2 Syntactic Features

This section first describes related work on syntactic features and than describes the type of syntactic feature developed for this research.

2.1 Related Work

A Word Sense Disambiguation (WSD)[1] system using syntactic features alone is described in (Lin, 2000). This system inspired the current research. This system also adhered to the principle of only using syntactic features **discovered automatically** in the corpus. In Lin's WSD system, one linked features were formed by starting at the target-word[2] and jumping to surrounding lexical nodes that could be reached from there. Two-link features were formed by jumping from the endpoint of one-link features to any non-target word nodes.

The precursor to this research, (Chanen and Patrick, 2004), details the use of ATSAS features for WSD. ATSAS features were based upon Lin's syntactic features but were designed to be more general. Lin's features were formed by expanding a feature one link at a time in a figurative line. The differences between Lin's syntactic features and those described in (Chanen and Patrick, 2004) are: 1) Lin's features are restricted to the linear (e.g. under his syntactic feature definition, a feature like Figure 1(e), where three links come together in one node, would not be possible); and 2) Lin's do not have abstraction.

A similar use of dependency parse subtrees is applied for producing state-of-the-art machine translation results in (Quirk et al., 2005), where the syntactic features are called *treelets*. After parsing a sentence in the source language, parallel treelets in the target language are then extracted to aid in the translation task. There is a very limited use of wildcards, only at the root of the tree.

For the experiments of this research, the syntactic features were extracted from dependency parse trees rather than from constituency parse trees. (Quirk et al., 2005) also used dependency parsers over constituency parsers citing the compactness of the sentence structure representation; for example, a verb is adjacent to its arguments in a dependency parse tree, whereas in a constituency parse tree the path from the verb to its arguments would first need to go up the tree before going down to the arguments.

Another WSD system using syntactic features is described in (Férnandez-Amorós, 2004). The system also uses parse subtrees as well as some wildcard abstraction for increasing recall, although his wildcards could only replace pronouns and content words. Much manual labour went into identifying the base syntactic features and the ways of abstracting the features. This research also uses

[1]See (Kilgarriff, 1998) for a description of the WSD task and the SENSEVAL competition.

[2]The word whose sense is being disambiguated

transformations on these sub-tree patterns in a further attempt to increase recall. This research did not achieve the same amount of automation in either identifying syntactic tree topologies or in generating wildcard features.

Not all syntactic features are directly related to the properties of a specific syntax tree. Some researchers have developed syntactic features that are aggregate, ratio, or statistical properties of all the sentences in a document or group of related documents. In one example of such syntactic features (Uzuner, 2005), the aim of the research is to differentiate between the works of different authors mainly by use of clever aggregate measures of parse tree properties. For instance, one set of features suggested in this research is the number of left-heavy, balanced, and right-heavy sentences that can be found in a particular author's work. By comparing averages and ratios of such features, powerful methods can be developed for differentiating between authors. These kinds of features have a different focus from ATSAS features, differentiating between authors. It might be interesting, though, to see if such aggregate features might be useful in a TC task like scam detection.

2.2 ATSAS Features

The type of syntactic feature that is developed here has been named the "All Topology, Semi-Abstract Syntactic" (ATSAS) feature type. This type of feature is extracted from a dependency parse tree. Dependency parse trees are preferred to constituency parse trees because they are more compact and arguably more directly related to document semantics (Schneider, 1998).

Each extracted feature is a subtree of a complete sentence parse tree. Theoretically, such features could consist of any number of links, but practically, a limit must be placed on the maximum number of links that are allowed to be in the extracted features. This is because problems with speed and memory limitations can result from the massive number of features that tree features have the potential to generate. With a maximum link level of only two, the experiments for this research generated over 10 million features. This massive number of features, though, was reduced by discarding features that occurred in less than three documents from a training corpus.

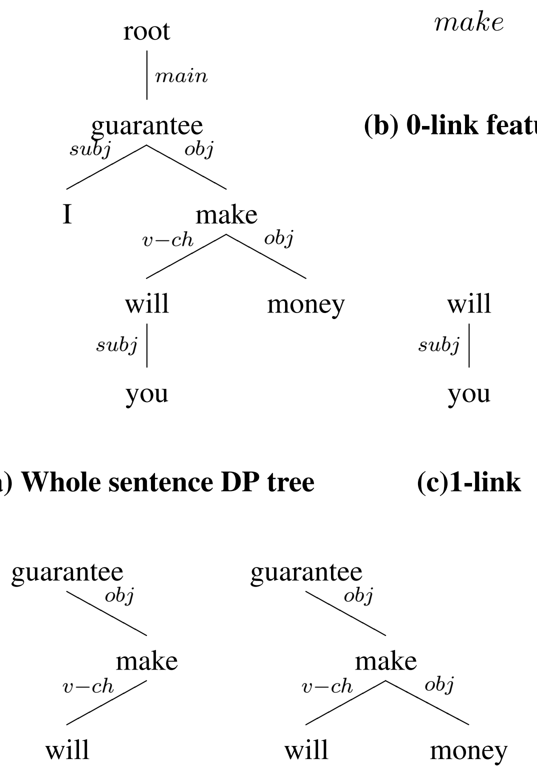
(Chanen and Patrick, 2004) succeeded in generalizing Lin's syntactic features to produce a greater variety of feature topologies, not just the linear features that Lin's system could extract. The current research had as a main goal of expanding the use of ATSAS features beyond their use in WSD to the TC task. In that main goal, this research seems the largely successful. However, the experiments in this research were not able to demonstrate the use of non-linear syntactic features such as the example seen in Figure 1(e) because of the memory limitations that occurred at feature extraction time. Since a maximum of two link features could be formed, the features extracted for this research is experiments were linear like those in Lin's system.

One of our important desideratum for syntactic features is that they are automatically extractable. Some syntactic features that were reviewed eariler in the Related Work section use manually-identified syntactic features. It is better, if possible, to eliminate human input in identifying the features both for greater efficiency and for the possibility that novel new syntactic features may be identified in places where human experts may not think to look or have time to look.

Figure 1 shows a small parse tree and several ATSAS features extracted from the complete parse tree with zero through three link example features.

Each feature link is labelled with a dependency link label e.g. *subj*, *obj*, *cc*. The links are joined by nodes that consist of both a lemma and part of speech (POS) tag. When the initial set of features is generated, all of the lemmas referred to literal sentence content. In a second phase of feature generation, all combinations of a feature's lemmas are abstracted to a "*". That is, each lemma in an ATSAS can be either literal or abstract. So, for a two-link, three-node feature, a total of seven abstract features can be generated from the one literal feature for a total of eight features. By such abstraction the hope is that some abstract features will join groups of documents in a way that is useful for the TC task. The nodes in Figure 1 do not show any abstraction and they also do not show the POS elements of the nodes.

One of the advantages of syntactic features like



(a) Whole sentence DP tree **(b) 0-link feature** **(c)1-link** **(d) 2-link feature** **(e) 3-link feature**

Figure 1: **(a)** The complete dependency parse tree of the sentence "I guarantee you will make money." **(b-e)** These figures show examples of syntactic features with zero (just a node) through 3 links.

ATSAS features is that it is possible to increase recall over features like n-grams because certain syntactic features will allow for matching parts of documents by skipping over some words. For instance, the syntactic pattern in Figure 1(c) can match, in a test document, the phrase "you will" as well as "you definitely will" even though "definitely" is not part of the syntactic pattern.

3 Methods

3.1 Data Parsing

Before extracting the syntactic features, the data was parsed using the Connexor dependency parser (Järvinen and Tapanainen, 1997).

3.2 Classifier

The maximum entropy classifier was selected to run the TC experiments, partly because publications such as (Ratnaparkhi, 1998) demonstrate that the ME classifier turned in state-of-the-art or near state-of-the-art results for a wide range of NLP tasks.

An important aspect of the ME algorithm is that it can robustly deal with a large number of features, which is important in this research since hundreds of thousands of features are possible (see (Ratnaparkhi, 1998).)

The software package used to perform the ME experiments of this research is the Megam (MEGA Model Optimisation)[3] package, which uses an optimised version of limited memory BFGS (Daumé, 2004).

3.3 Training and Testing

Ten-fold cross validation was used for the training and testing evaluation process. All tests used the same set of folds.

For any given training fold, after all the given feature types were extracted from the train documents, all token types were removed from consideration if that token type was observed in less than three documents.

Classification accuracy was chosen as the performance metric in these experiments.

Hypothesis testing was used to determine if the TC performance between a pair of feature types was significantly different. The paired t-test was the specific type of hypothesis test used. A standard significant cutoff value of $\alpha = 0.05$ was utilised. The null hypothesis is that the 10-fold cross validation determined accuracies of any two feature tests do not differ significantly from each other.

4 Experiments

This section describes the series of experiments performed to determine the usefulness of the ATSAS family of syntactic features.

[3]This package is described and can be downloaded at the website: `http://www.cs.utah.edu/∼hal/megam/`

Feature or feature combination	
Title	**Description**
SAA	Only the ATSAS features that have only abstract lemmas. "AA" stands for **all abstract**. Every lemma in this feature set is abstacted to a "*".
2g	The corpus is represented only in terms of bigrams.
1g	Representation only in unigrams (bag-of-words). The baseline feature.
1g2g	A combination of unigrams and bigrams
1gS-A	A combination of ATSAS and unigrams features except for any syntactic features that contain any abstract lemmas. Here "-A" stands for **minus abstract**.
S-A	All ATSAS features except for any syntactic features that containing any abstract lemmas. In other words, this is the set of literal syntactic features.
S	All ATSAS features Including abstract and non-abstract features.
S-NA	All ATSAS features except for any syntactic features that do not have abstract lemmas at all. Here "-NA" stands for **minus non-abstract** where non-abstract means every feature in which no lemmas are a "*". In other words, this is the S set minus the totally literal features.
1gS-NA	All ATSAS and unigram features except for any syntactic features that do not have any abstract lemmas at all.
1gS	A combination of unigram features and all ATSAS features

Table 1: Descriptions of the feature types used in this paper's experiments.

4.1 Data

A subset of the original ScamSeek (Patrick, 2006a) project's dataset was used in this research. This subset consists of 2130 documents. This means that during 10-fold cross validation, each training fold consists of 1917 documents and each testing fold consists of 213 test documents. Out of the total corpus, 27% of the documents are scams and 73% are non-scams.

4.2 Features

A variety of different feature types are compared in the below experiments. Table 1 gives the abbreviated name and feature description for each feature type (or a combination of feature types)associated with one of the experiments.

Table 2 shows the number of token types in the **S** and **S-A** feature types, broken down by the number of links in each feature type. This table illustrates how the abstraction of some elements of literal syntactic features helps multiply the number of statistically significant ATSAS features. At first glance, the feature counts for the **S-A** (literal or non-abstract) feature type seem to be counterintuitive. One might expect an explosion of features as more links are added. However, since features observed in less than three documents in the training set are removed, this causes the number of significant 2-link **S-A** features to be less than the number of 1-link **S-A** features. When one or more of the lemmas in an ATSAS feature are abstracted, two or more literal features that were distinct may map to the same abstract feature. Literal features that are not significant on their own, may produce a significant feature through this abstraction process. The count of an abstract feature is the sum of the counts of the literal features that map to it. This is the reason for the much higher number of statistically viable features in the 2-link **S** feature type, where partial or full abstraction of lemma elements is allowed.

Producing abstract features can be quite memory intensive because all literal features found in any document must be stored even if a feature occurs in just a single document. This is necessary since only after all documents have been processed can the abstraction process join low count literal features. The full **S** feature type includes more than 10 million features after all abstract features have been added to the literal features. This exponential explosion of features is the reason for not being able to perform any TC experiments using 3-link features. See the Future Work section for ideas for including 3-link features.

	Non-abstract (S-A)		**Abs. and Non-abs. (S)**	
	count	**%**	**count**	**%**
0-links	7873	10.6%	7873	2.2%
1-link	34466	46.4%	70610	19.8%
2-link	31895	43.0%	277301	77.9%
Total	**74234**		**355787**	

Table 2: Feature counts in both the **S-A** experimental set (which includes only features with no abstraction) and the full set of ATSAS features in **S** (which includes both abstract and non-abstract syntactic features) as the number of links are inceased. A 0-link feature is just a lemma/POS node.

5 Results

The results of the experiments are measured in terms of the classification accuracies of each type of feature, as well as the pairwise t-test results for each pair of feature experiments. If the t-test results for a given pair is at or below the $\alpha = 0.05$ level, then it is assumed that results for such a pair of experiments will differ significantly. Table 3 displays the results. This table presents the number of features in each feature set and also gives the performance of each feature type in terms of classification accuracy. The table also gives the paired t-test score for each pair of experimental results. The t-test pairwise significance results are displayed in the half-triangle part of the table where significant results are shown in bold face.

6 Analysis

The baseline for these tests is the Results using the bag-of-words feature, **1g**, which shows an accuracy of 86.9%. In order to judge the ATSAS feature type (or some derivative thereof) as a successful TC feature, we need to show a significant t-test score between one of the ATSAS feature types and the bag-of-words features.

Experiment one used ATSAS features that eliminated all features that had literal lemmas (i.e. not abstract). There are only 4,291 instances of this subtype of ATSAS features. Not surprisingly, this feature type did the worst with an accuracy of 82%, which was significantly worse than every other type of feature tested. One hypothesis as to why this feature type did not perform well is because there are too few features to perform good classification alone. Another possibility is that the literal lemmas in many of the ATSAS features are important and help the best of these features two separate the classes. In other words, having some literal content may be helpful for forming good features for separating classes.

Another non-syntactic feature that was experimented with, was a pure bigram (2g) feature representation. See experiment #2 in Table 3 for the results. One can observe that the bigram representation scores significantly below the unigrams representation of the corpus in the TC task. Even though bigrams often carry more meaning than a

Experiment		**Corpus features**	**Score**	SAA	2g	1g	1g2g	1gS-A	S-A	S	S-NA	1gS-NA
Title	**#**			1	2	3	4	5	6	7	8	9
SAA	1	4,291	82.0									
2g	2	141,346	85.5	**0.0024**								
1g	3	20,402	86.9	**0.0001**	**0.0372**							
1g2g	4	161,748	87.0	**0.0001**	**0.0341**	0.8456						
1gS-A	5	94,576	88.6	**0.0013**	**0.0239**	0.1830	0.1497					
S-A	6	74,234	88.8	**0.0000**	**0.0004**	**0.0003**	**0.0048**	0.8845				
S	7	355,787	89.1	**0.0004**	**0.0078**	0.0709	0.0689	0.5797	0.8501			
S-NA	8	281,553	89.2	**0.0000**	**0.0008**	**0.0171**	**0.0051**	0.6334	0.6249	0.9341		
1gS-NA	9	301,895	89.3	**0.0002**	**0.0043**	**0.0419**	**0.0365**	0.3859	0.6899	0.1382	0.8921	
1gS	10	376,129	89.4	**0.0002**	**0.0020**	**0.0288**	**0.0240**	0.2695	0.6135	0.1530	0.8138	0.5911

Table 3: Experimental results with significance of paired t-tests

single word, because of sparsity problems with word combinations, bigrams usually fall short of unigrams. A combination of unigram and bigram feature types scores slightly higher than unigrams alone, but not significantly so. The bigrams feature is one of the simplest complex features. This experiment demonstrates how even simpler complex features suffer from sparsity.

For the **S** feature type (FT) (Feature type 7 in Table 3), although the accuracy is 2.2% higher then the bag-of-words features, the t-test is not significant (although it is close to being significant). However, if the **S** feature type is combined with the **1g** feature type then the results are a full 2.5% better in terms of accuracy and are significant (See row 10 in Table 3.)

Another positive result for the ATSAS family of syntactic features can be observed for FT **S-A** (#6 in the results table). This result is positive in that it is the most significant t-test score with the bag-of-words feature when compared with all the other feature types and their t-test score with the **1g** FT. This is surprising for at least two reasons: 1) one of the initial assumptions for including partial abstraction as a property of ATSAS features was that the abstraction would alleviate problems with sparsity and the syntactic features would not do well without it; and 2) the **S-A** FT has only 74,234 token types from the whole corpus compared to 355,787 token types for the **S** FT. One possibility is that some feature selection would be beneficial even though publications such as (Ratnaparkhi, 1998) claim that the maximum entropy classification method does not suffer much from the curse of dimensionality. Another thought is that, because tens of thousands of non-abstract features are generated as the sentence parse trees of documents are traversed across the entire training set, many good TC features are discovered even though they may not take advantage of the power of abstraction.

The results from experiment #6 suggest that abstraction is not necessarily needed for ATSAS features to realise gains over simple bag-of-words. However, we do have some evidence that abstraction in ATSAS features does help them achieve even better performance. Experiments 8 and 9 involving the feature types **S-NA** and **1gS-NA** respectively, and both have 10-fold cross validation accuracies that allow the rejection of the null hypothesis. Therefore, it can be concluded that these feature types are significantly better than bag-of-words alone. These two feature types do better then experiment #6 involving the **S-A** feature type, by an accuracy difference of 0.4% and 0.5% respectively. Unfortunately, the difference between the three feature types **S-A**, **S-NA**, and **1gS-NA** is not significant so further experimentation would be required to settle the question of whether abstraction is needed more satisfactorily.

7 Conclusion

More complex feature types may seem desirable because they may allow for features that are closer to the semantic level and thus possibly better for difficult TC tasks. However, experience has often shown that more complex features do not usually live up to their promise because of sparsity problems.

In this research we have proposed a family of syntactic features, the All-Topology, Semi-Abstract Syntactic feature family. We have experimentally shown that several variations of ATSAS feature types have significantly out-performed bag-of-words features. Specifically, the set of subtrees from a dependency parse tree with zero through two links when combined with bag-of-words gave the best performance, significantly better than bag-of-words alone. Surprisingly, variations on ATSAS that either eliminated abstract features or eliminated totally literal features also did significantly better than bag-of-words alone.

8 Future Work

A logical next direction would be to expand the ATSAS by pushing the maximum number of links per feature from two to three. With only two links, the syntactic features are linear just as in (Lin, 2000). With three or more links, the kinds of tree topologies and the different combinations of dependency link types would increase exponentially along with greater possibilities of finding an interesting syntactic feature for separating classes.

However, memory limitations prevented using three links as the maximum. This problem could be addressed in several ways. As experiment #6 involving the **S-A** variation suggested, ATSAS features do not necessarily need abstraction to perform well. So a logical step is to simply see if there is enough memory if three link ATSAS are generated without any abstraction.

Another direction is to only generate ATSAS features for TC that involve certain lemmas. For instance, the top N ($N < 500$) words by information gain or some other measure could be used. If an ATSAS is generated but does not involve one of the selected lemmas, then that feature would be discarded. Another way to determine the top terms from which the syntactic features would be built would be to take the top terms by probability from each topic from a Latent Dirichlet Allocation (LDA) (Blei, 2004) model. The topical terms might differ from the information-gain terms in interesting ways.

It would also be desirable to see if similar, encouraging results can be derived from using the same type of features on public-domain and widely used benchmark datasets such as Reuters.

Finally, using a better feature selection strategy might be advantageous. (Ratnaparkhi, 1998) presents evidence that the maximum entropy learning algorithm can handle a large number of features with little degradation in performance. However, that evidence was based on a single dataset. It is possible that the ScamSeek dataset might benefit from a more sophisticated feature selection strategy. Feature selection might also help separate the truly useful ATSAS features from the background noise.

Acknowledgements

We would like to thank the Capital Markets CRC and the University of Sydney for financial support and everyone in the Sydney Language Technology Research Group for their support. Also, thanks to Sue Zeckendorf for editing help .

References

[Blei2004] David Blei. 2004. *Probabilistic models of text and images*. Ph.D. thesis, U.C. Berkeley.

[Chanen and Patrick2004] Ari Chanen and Jon Patrick. 2004. Complex, corpus-driven, syntactic features for word sense disambiguation. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 1–8, Sydney, Australia, December.

[Daumé2004] Hal Daumé. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper and implementation available at `http://www.cs.utah.edu/~hal/megam/`, August.

[Férnandez-Amorós2004] David Férnandez-Amorós. 2004. Wsd based on mutual information and syntactic patterns. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 117–120, Barcelona, Spain, July. Association for Computational Linguistics.

[Järvinen and Tapanainen1997] Timo Järvinen and Pasi Tapanainen. 1997. A dependency parser for English. Technical Report TR-1, Department of General Linguistics, University of Helsinki, Finland.

[Kilgarriff1998] 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.

[Lewis1992] David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50. ACM Press.

[Lin2000] Dekang Lin. 2000. Word sense disambiguation with a similarity-smoothed case library.

[Moschitti and Basili2004] Alessandro Moschitti and Roberto Basili. 2004. *Complex Linguistic Features for Text Classification: A Comprehensive Study*. Springer Verlag.

[Patrick2006a] Jon Patrick. 2006a. The scamseek project - text mining for financial scams on the internet. In *Selected Papers from AusDM*, pages 295–302.

[Patrick2006b] Jon Patrick. 2006b. The scamseek project: Text mining for financial scams on the internet. In Graham J. Williams and Simeon J. Simoff, editors, *Selected Papers from AusDM*, volume 3755 of *Lecture Notes in Computer Science*, pages 295–302. Springer.

[Quirk et al.2005] Christopher Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *ACL*. The Association for Computer Linguistics.

[Ratnaparkhi1998] A. Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

[Schneider1998] Gerold Schneider. 1998. A linguistic comparison of constituency, dependency and link grammar. Technical Report 1.1, Institut für Informatik der Universität Zürich.

[Uzuner2005] Özlem Uzuner. 2005. *Identifying Expression Fingerprints Using Linguistic Information*. Ph.D. thesis, MIT.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 28-36

Lexical Access via Phoneme to Grapheme conversion

[1]**Michael Fridkin,** [1]**David L. Dowe and** [2]**Simon Musgrave**
[1]Clayton School of I.T
[2]School of Languages, Cultures and Linguistics
Monash University
VIC 3800, Australia
mfri7@student.monash.edu.au
david.dowe@infotech.monash.edu.au
simon.musgrave@arts.monash.edu.au

Abstract

The Lexical Access (LA) problem in Computer Science aims to match a phoneme sequence produced by the user to a correctly spelled word in a lexicon, with minimal human intervention and in a short amount of time. Lexical Access is useful in the case where the user knows the spoken form of a word but cannot guess its written form or where the users best guess is inappropriate for look-up in a standard dictionary or by traditional computer spellcheckers. Previous approaches to this problem have attempted to match user-generated phoneme sequences to phoneme sequences in dictionary entries and then output the stored spelling for the sequences. Our approach includes a phoneme-to-grapheme conversion step followed by spelling-to-spelling alignment. This ensures that more distortion to the data can be handled as well as reducing error rates and also allows us to combine the Lexical Access problem with a related task in Natural Language Processing, that of spelling Out of Vocabulary (OOV) items. We call this combination the Hybrid Lexical Access (HLA) Problem. The performance of our system on this task is assessed by the percentage of correct word matches in the output after edit distance is performed to find the closest match to a stored item.

1 Introduction

In languages with irregular spelling systems, such as English, most speakers have trouble spelling at least some words. To aid spelling there are orthographic dictionaries (which index words by their sequence of letters) and phonemic dictionaries (which index words by their sequence of phonemes). We will be looking at how, given a phoneme sequence, phonemic dictionaries can be used by computer systems to look up single words and if the word is not in the lexicon then how computer systems can best give a suggestion to its true spelling.

In orthographic dictionaries, finding the correct word depended on being able to spell the first few letters correctly. Spellcheckers have improved on this method but still have two drawbacks. Firstly, they require a candidate spelling before they can begin matching and this can prove difficult in some situations; for example, an unstressed vowel sound (normally schwa in English) can be spelled with many different graphemes. Secondly, spell-checkers have difficultly matching multiple graphemes which can correspond to the same phoneme. For example, in the word Aachen, the candidates Acen, Aken and Acken would not map to the correct spelling using standard spellcheckers. Few speakers of English would know that /k/ is spelled as *ch* in some circumstances such as the one in the example. Computer systems can supplement the user's linguistic knowledge in these situations.

In Computer Science the *Lexical Access Problem* involves matching a sequence of phonemes to a sequence of words. It is assumed that each phonemic sequence corresponds to some word in the lexicon and it is the job of the computer system to ascertain what the intended word is. We wish to do a similar task but for single words and without the assumption that the word is in the dataset. If the word is not in the dataset then the problem turns into that of spelling Out of Vocabulary (OOV) items. Obviously the user can not specify if the word is or is not in the dataset and the system must make that judgement itself. This extension of the Lexical Access Problem to include OOV items we call the Hybrid Lexical Access (HLA) Problem. Besides improving lookup of spelling for unfamiliar and unknown words, both aspects of this hybrid problem are also relevant as part of automatic speech recognition systems. The focus of this article is to compare and contrast the Lexical Access, Spelling OOV items and HLA problems and their applications to aid people with spelling.

2 Background

The dataset for the HLA problem is one or more phonemic dictionaries (the problem is restricted to single words so continuous speech data is irrelevant). Since dictionaries are transcribed by expert linguists or by automatically utilizing linguistic rules made by linguists and users are not expected to have the same training, there is no guarantee that both will agree on a correct pronunciation. The system must assume that most times there will be disagreement. The main role of the system is to standardize input from users to match, ideally, a single dictionary entry.

The system is given 1 attempt to produce the correct match for the input during testing but in practice it is allowed to output as many words as necessary (to cover the case of multiple homophones - i.e. to, two and too). We need to measure the system's accuracy by how accurately it performs across a wide range of words, to prevent bias towards one class of words, such as short words or common words or words with silent letters.

An example of a flaw in the system is if a dataset has silent letters, such as "w" in wrong, aligned to a special null or empty phoneme resulting in silent letters being second-class citizens and likewise the words containing the silent letters. If the "w" in wrong and "p" in psychology are treated as the same phoneme and there are 25 graphemes which may correspond to this phoneme in total then each will have a base probability of roughly $\frac{1}{25} = 4\%$. Such a system will have a high accuracy because it will give the correct output for the other phonemes. The output overall will be very close to being correct. The problem is that the bias will be directed towards words without silent letters. Those words that do not belong to this class will be significantly difficult for the system to find, roughly 25 attempts on average to find the "w" in wrong, which is not practically feasible.

Our system first does phoneme-to-grapheme conversion (PTGC) and then finds the best match from the dataset for that grapheme sequence (using the ASPELL spellchecker). If no match is found, then the generated grapheme sequence is the final output of the system. The metric which we use for measuring the success of the first stage of the system is a Average Conversion Percentage (ACP) at word level, corresponding to what (Marchand and Damper, 2000) refer to as "percent correct" and what (Rentzepopoulous and Kokkinakis, 1996) call "success rate of the conversion algorithm ... at the word (state sequence) level". The ACP is the percentage of entries converted correctly in the PTGC stage out of the total number of entries produced. No edit distance measure is included at this stage. During post-processing ASPELL matches the grapheme string output from the PTGC to a word in the dataset using the Minimum Edit Distance criterion which selects the grapheme representation for which the minimum number of insertions, deletions and substitutions is required (Levenstein, 1966). The success of the system at this stage is measured using ACP Average Conversion Percentage after Edit Distance (ACP-Ed), which again represents the percentage of correct matches out of the total number of outputs produced.

ACP-Ed was used to control training of the PTGC component and it has several advantages for this purpose. Firstly, it reduces any bias in the assessment process towards short words and common words. If Edit Distance is not taken into account, the performance of a system may be exaggerated if it handles such words well even if its output when dealing with longer words and uncommon words is very poor. Using ACP-Ed means that the system is guided towards producing good output for all input, even if very few inputs achieve perfect output. Secondly, ACP-Ed takes into account the size and distinctiveness of the available dataset and assesses not only how close the output of PTGC is to the correct graphemic representation, but also how close the output is to similar dictionary entries and whether sufficient information is available to discriminate between similar entries. For example, the phonemic input [a b d uh k t @r r z] (‘abductors’), produces ‘abducters’ as output from PTGC. This has an Edit Distance of 1 from the correct graphemic representation (substitution of ‘o’ for ‘e’ is required at position 7), which is a good result. However, the dictionary also contains the entry ‘abductees’, which also has an Edit Distance of 1 from the PTGC output (substitution of ‘e’ for ‘r’ is required at position 8). Therefore, the PTGC output is not good enough to disambiguate between the two possible matches with grapheme strings from the dictionary and the result will count as a failure in the calculation of ACP-Ed.

When plotted on different axes, we see fluctuations between ACP and ACPed, indicating a non-monotonic relationship. Indeed, these fluctuations occur with some consistency across the two data-sets (UNISYN and *NETtalk*) used in our study and again when we varied the internal organisation of the PTGC process - namely, the format table encoding from sec. 5.4.

3 Previous Work

Pronunciation by analogy (Marchand and Damper, 2000) is a PTGC technique which matches substrings of the input phoneme sequence to substrings of a group of words with similar phoneme sequences in the dataset and uses their corresponding grapheme sequences to infer the output grapheme sequence. The system of that study used the *NETtalk* corpus of approximately 20,008 hand-aligned entries with stress and syllable boundaries to achieve 76.4% ACP.

NETtalk was produced by Terrence J. Sejnowski and intended for grapheme-to-phoneme conversion, it is setup using 50 phonemes and a null phoneme for silent letters. In practical situations the null phoneme, which gives cues to the system about locations of silent letters, is not given to the system. This makes this dataset an interesting area of investigation for phoneme-to-grapheme systems because accurately predicting how the null phoneme is spelled would greatly improve accuracy for silent letters. Roughly half of the *NETtalk* dataset contains the null phoneme. This is due to the fact that vowels can be monophthongs such as “e” in pet or dipthongs as in “a” in pay. The word pay is transcribed /pe-/ where the “e” stands for /eɪ/ and the y corresponds to the null phoneme. To make the dataset useful for training and testing PTGC systems, an auxiliary system needs to be created which takes as input a phonemic sequence and inserts the null phoneme in appropriate locations.

A run of the Snob (Wallace and Dowe, 2000) software on the *NETtalk* dataset revealed that roughly 70% of the time when the null phoneme was observed it was surrounded by predictable neighbours or one of several predictable neighbours. For example, the phoneme /eɪ/ represented in *NETtalk* by the letter “e”, as a first vowel in the nucleus of a syllable receiving secondary stress in penultimate position, occurs 64 times in *NETtalk*. Out of those 64 times, 60 times the null phoneme is at the last position. To be even more precise, when /eɪ/ is spelled using the letter “a” and is the first vowel in the nucleus of a syllable receiving secondary stress in penultimate position it occurs 52 times in the dataset, all of which are followed by the null phoneme in last position. The predictability of the null phoneme can be the basis for a system for converting data in *NETtalk* format into one that is compatible with the general format of input to PTGC systems (which does not have the null phoneme).

The approach in (Thomas et al., 1997) to the Lexical Access problem was to match the input phonemic sequence to a phoneme sequence from the dataset and then output the corresponding grapheme sequence. Their dataset contained more than one realization of the same word so they used edit distance to match the input sequence to one of the realizations seen in the dataset from which the corresponding grapheme sequence was the output. The testing was done by inputting phoneme sequences and measuring how many input phonemes differ from a candidate stored lexicon sequence divided by the to-

VB - *Verbs*		
VBD	VBG	VBN
past tense	gerund	past participle
JJ - *Adjectives*		
JJR	JJS	
comparative	superlative	
NN - *Nouns*		
NNS	NNP	NNPS
plural	proper	proper plural

Table 1: Part of speech (POS) abbreviations

Orthography	a	b	o	a	r	d
NETtalk	x0	b	o1	-⟨	r	d
IPA	ˌɑ	ˈb	ɔː		ɹ	d
Unisyn	@	b	*oo	a		rd
IPA	ə	ˈb	ɔː			ɹd

Table 2: Comparison of dataset representations of the word *aboard* with different alignment strategies.

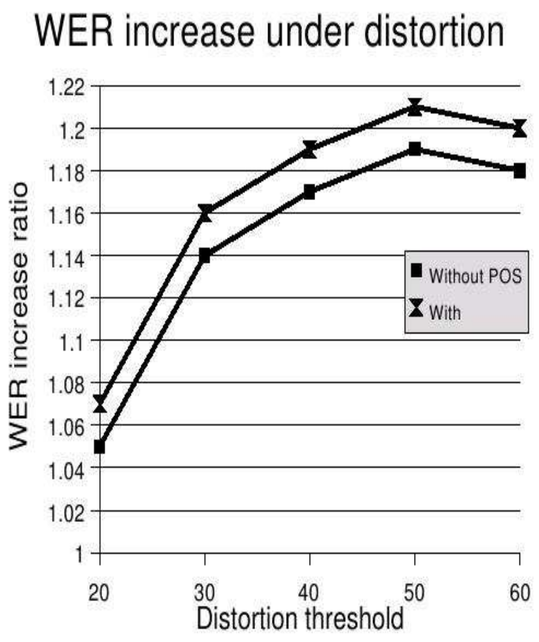


Figure 1: (Thomas et al., 1997) Results

tal number of phonemes, called the distortion rate. The distortion threshold is the maximum distortion rate for which the corresponding average Word Error Rate (WER) was calculated. With Part of speech (POS) tags in the input they achieved results of 24.53% WER under 10% distortion threshold and without POS tags, 31.29%. The degradation in performance of their system, with POS tagging and without, when increasing the distortion threshold can be seen in figure 1.

Two points of interest in figure 1 are that the last increase in distortion from 50% to 60% actually decreases the WER by 0.19% and that the model with POS tagging show a higher degradation under distortion than the one without POS tagging, which corresponds to our results for POS tagging (sec. 6). For example, when increasing the distortion rate from 10% to 20% the WER of the model with POS tagging increased by 1.62% whereas the model without POS tagging, with the same increase in rate of distortion increased by 1.56%. As a ratio of the original 10% WER, these come to 1.07% and 1.05% respectively as seen in figure 1, for 20% distortion. The 0.02% difference between the two models stays roughly constant as the distortion ratio increases.

4 Resources Used

There were two datasets used for training and testing our system, the UNISYN *Accent-Independent Keyword Lexicon* and the *NETtalk* corpus. The two datasets differed both in size and the levels of detail. The UNISYN dataset had roughly 119,356 entries, after entries with special symbols such as apostrophes were removed. Each entry contains orthography, POS tag, phonemic transcription, syllable boundaries, morpheme boundaries and typical word frequency as found in standard literature. However phoneme to grapheme alignment is not included and had to be done separately. The alignment strategy for this dataset, A_U, partitioned the dataset into phoneme and grapheme pairs using a hand-made table of intuitive correspondences.

The *NETtalk* corpus has for each entry an orthography, phonemic transcription and an indicator variable in the range 0 to 2 to say whether the word is irregular, foreign or common. The phonemes and graphemes are aligned one-to-one in a format suitable for grapheme-to-phoneme conversion. There is a null phoneme which corresponds only to silent letters and makes the correspondences more regular for the other phonemes.

The *NETtalk* alignment strategy we call A_N. A comparison of the results of the two models of alignment is in table 2.

To prepare the datasets for training and testing, lexical stress, POS tags and phonemic transcriptions were extracted from them and an alignment strategy was applied. For the *NETtalk* dataset the POS tags were matched to the UNISYN dataset where they had entries in common and the rest of the tags were automatically generated using *C&C Tools* (Clark and Curran, 2004).

For analyzing the datasets the Snob program was used, which does flat clustering using the Minimum Message Length (MML) criterion. When a dataset is run through Snob using an alignment model, A_N or A_U, the output is a set of classes all at the same depth for each pairing in the model. The classes are a hypothesis about the data that when applied to the data and transmitted with the data as a message has the shortest two-part message length (Wallace, 2005) (Wallace and Boulton, 1968) (Patrick, 1991). The premise is that the shortest message Snob finds will be a good explanation of the data.

5 Method

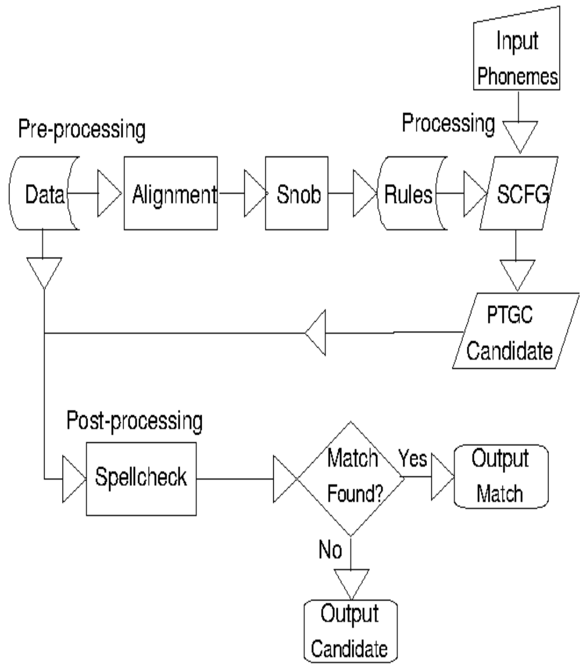


Figure 2: System flow

The system can be divided into 3 modules: (i) Pre-processing, represented in figure 2 as the steps from "Data" to "Rules", (ii) Processing, starting at the "SCFG" step and ending at the "PTGC Candidate" step, and (iii) Post-Processing, starting at the step "Spellcheck" and ending with either "Output Candidate" or "Output Match".

The pre-processing stage aligns the dataset by using an alignment model if the dataset is not already aligned and generates a list of classes produced by running Snob on the aligned dataset. The list of classes are passed to the parser to be converted into a Stochastic Context Free Grammar (SCFG).

The processing stage of the system is given the input in the form of a phoneme sequence and its POS tag, as well as a SCFG. Each phoneme in the sequence is processed sequentially and its output graphemes are concatenated to produce the candidate word.

The post-processing stage is given the candidate word with the dataset only containing the word spellings and it has to find the best match or fail and output the candidate word and part of speech tag it was given.

5.1 Pre-processing

The dataset usually consists of a series of entries of phoneme sequences and word spelling. In the phoneme sequence each phoneme is written out separately and the word spelling is written together as a single string. We want to look at the structure of the word to create models for understanding the relationship between the phoneme sequence and the grapheme sequence and this requires breaking up the word spelling into a grapheme sequence with each member of the grapheme sequence having one or more graphemes corresponding to one phoneme.

Phonemes with different levels of stress are considered as different, as well as phonemes that are the beginning of names. In total there are 657 pairs, 463 part of speech tag combinations (35 basic tags) - which are only used at the post-processing "Spellcheck" step - and 157 phonemes (including merged phonemes).

The dataset now consists of a series of pairs of phonemes and graphemes and each possible pairing is given a unique ID. A phoneme can have multiple grapheme pairings so it spans over several ID's. We choose the ID's for each pairing carefully following the criteria that for each phoneme the ID's which correspond to that phoneme are contiguous, for example {39 k_c, 40 k_ck, 41 k_k}. ID's in each possible broad sound groups (such as monophthong vowel, diphthong vowel, stop or fricative) are also contiguous.

We can now view each ID in the list as a separate entity, distinct from the word in which it was originally found, surrounded by zero or more neighbours and having a position in the ID sequence. This list of independent ID's is used as input to the Snob program in order to generate classes which use patterns in the occurrence of neighbours to compress the dataset into groups whose combined size equals a fraction of the size of the original dataset.

Groups are chosen following two criteria. Firstly, the ID in any group is exactly the same or very close (for example, a group may consist of a vowel and a stressed form of the same vowel), and secondly the neighbours and position for that ID are as different as possible to all the other ID's which have the same phoneme (we do not need to worry about ID's that do not have the same phoneme because in the input we are given the phoneme). Each group is described by a Normal distribution for each of four neighbours (neighbour two spaces to the left to the neighbour two spaces to the right, and position of the ID). We end up with a distribution with the mean centered not at the most frequent ID but at the ID which has a value which is in between the most frequent ID's. These are our classes which from now on we refer to as rules.

5.2 Processing

The system parses an input phoneme sequence by looking at the environment (its neighbours and position) of each of the phonemes and looks up every relevant rule in the SCFG to find the most likely rule. All the rules we select have a distribution centered at one particular value for attribute 6 which is outputed by the parser when the rule is selected.

For example, for the pair "e ə" (meaning schwa is spelled with an "e" having ID 63 in our model (there are 1031 such pairs in a sample dataset of 10,000 words, as in "abhorr*e*nce". Snob will find a class with this identical spelling of schwa and the description of the attributes of this class will inform us when to use this spelling. This is the class which describes this pair:

```
Serial 517 Relab 0.019 Size 152.
        Att 2 vals 152.
              Mean 508.5465 S.D. 66.9917
        Att 3 vals 152.
              Mean 469.9997 S.D. 0.4144
        Att 4 vals 152.
              Mean 502.1750 S.D. 88.1242
        Att 5 vals 152.
              Mean 5.9335 S.D. 2.2393
        Att 6 vals 152.
              Mean 63.0000 S.D. 0.0400
```

Att1 is the model for the pair 2 spaces left, if there is none then the value is -1.0, likewise Att2 is the pair 1 space to the left, Att3 is the pair 1 space to the right, Att4 is the pair 3 spaces to the right and Att5 is the position of the ID for the current class. Most importantly att6 is the description of all the ID's in this class. The standard deviation of 0.04 means that all the ID's are exactly the same. Although there are 6 attributes Snob looked at, it chose the above 5 for this class and attribute 1 it deemed as insignificant for this class. Snob deems attributes as insignificant when their description is almost identical to the description of the population (for this attribute it is [Mean 246.7137, S.D. 210.8832]). Our current population is all possible pairings of schwa with any graphemes. To interpret this class we need to look at the range of the different Broad Sound Groups (BSG).

Start marker	-1	End marker	629
Monophthong	1 .. 215	Dipthong	216 .. 394
Tripthong	395 .. 404	Plosive	405 .. 454
Nasal	455 .. 494	Approximant	495 .. 546
Fricative	547 .. 574	Affricate	575 .. 618
Trill	619 .. 628		

This class says that we spell schwa as "e" when it is around position six preceded by a Plosive or Approximant and followed by "n" and then by a Plosive, Approximant or Fricative. These are the combinations "ment", "dence" and "tence". In our example the neighbours of the schwa are "ˈo r" on the left and "n s" on the right, that is Att1 is in the range 164 to 167, Att2 is 619 to 629, Att3 is 460 to 476 and Att4 is 559 to 574. All of the values lie without one standard deviation from where the class predicts except for Att2 which is two standard deviations away. This is still a high probability because 68.3% of the values lie within one standard deviation of the mean for the Normal distribution and 95.5% lie within two standard deviations.

In order to automatically find this rule we write all the classes we are interested in - that is all the classes that have an ID with the schwa phoneme - as production rules in the SCFG with attributes one to five on the right hand side and attribute six on the left. Next we give each class a weight which reflects the size of this class divided by the total size of the population. Then we give each input symbol a weight which reflects the probability of seeing this symbol when looking at the appropriate attribute for all the members of this class. Finally we choose the rule which has the highest probability of being parsed. A fragment of the SCFG for class 517 given the input “{ ˈo r ə n s 6 }” is:

1.0	Start	–>	St
0.15	e	–>	{ Att1 Att2 ə Att3 Att4 Att5 }
0.0017	Att1	–>	ˈo
0.0013	Att2	–>	r
0.0618	Att3	–>	n
0.0034	Att4	–>	s
0.1780	Att5	–>	6
0.13	St	–>	e

Non-terminals = {St, Att1, Att2, Att3, Att4, Att5}, start Symbol = {Start}, terminals = {e, ˈo, r, n, s, 6, {, }}. The weights for the St symbol represent the relative frequency of the ID in relation to the other ID’s of this phoneme, the weight for the rule (surrounded by curly braces) is the size of the group it represents as a fraction of all the ID’s of this phoneme and the weights of each attribute is calculated by sampling over the range of the input phoneme corresponding to the appropriate attribute. We sample from the Normal distribution which the attribute models, over the range of the input divided by the magnitude of the range to calculate the average probability of each ID which the input ranges over of belonging to this class. For example, attribute 1 is N(μ = 246.7137, σ = 210.8832) and the first input is “ˈo” with range 164 to 167 so the weight in the SCFG for this attribute given the current input is 0.0017.

Formally a SCFG is a pair (p, G) where: $p : P \rightarrow [0, 1]$ is a probability function assigning probabilities to rules such that $\sum_{i \in \Pi_A} p(A \rightarrow \alpha_i) = 1$ and Π_A is the set of all rules associated to A, (Benedí and Sánchez, 2007). A SCFG is consistent if the probability assigned to all sentences in the language of the grammar sums to 1.0.

The result of a Snob run (Wallace and Dowe, 2000) using the six attributes discussed at the start of the section was a set of classes, for each pairing of phoneme to graphemes, with each class having a model containing the mean, standard deviation and relative abundance. This produced a SCFG consisting of 35,000 classes over a dataset of 99,800 entries and 657 pairs, with an average of 53.27 classes needed to describe each ID.

Sys	**Data**	**Size**	**ACP** [a]	**ACP**[b]	**ACP**[c]
A_U[d]	UNI[e]	10K	16.53		84.74
A_N[f]	NET[g]	20K	31.27		91.00
A_N	NET[h]	20K	31.27		81.20
M.[i]	NET	19.5K		75.1	

[a]ACP when the system has no access to the lexicon
[b]ACP when the system has access to all but the target entry of the lexicon
[c]ACP when the system has full access to the lexicon, i.e. ACPed
[d]Our system using UNISYN alignment
[e]UNISYN dataset
[f]Our system using NETtalk alignment
[g]POS tags added to NETtalk
[h]NETtalk dataset
[i](Marchand and Damper, 2000) under 0% distortion

Table 3: Comparison of Systems

From,To	**[ə, æ]**	**[aɪ, æ]**
Effect	-0.73%	-0.17%
Freq.	46.0%	6.78%

Table 4: Effects of Vowel Distortion

5.3 Post-processing

After a candidate is generated its most common part of speech tag is attached to it and an attempt is made to match the candidate to a lexicon word.

From,To	[d,ʃ]	[ʧ, ʃ]	[t, ʃ]
Effect	-3.11%	-1.21%	+2.39
Freq.	23.31%	3.05%	7.8%

Table 5: Effects of Consonant Distortian

From, To	NNP, NN	NNPS, NN	NNS, NN
Effect.	-1.85%	-0.76%	-3.7%
Freq.	14.4%	1.5%	18.0%

Table 6: POS distortion - Nouns

From, To	VBD, VB	VBG, VB	VBN, VB
Effect	-1.51%	-0.24%	-1.08%
Freq.	3.52%	6.7%	2.96%

Table 7: POS distortion - Verbs

From, To	JJR, JJ	JJS, JJ
Effect	-0.35%	-0.13%
Freq.	0.5%	0.71%

Table 8: POS distortion - Adjectives

If the candidate cannot be matched then the output is the candidate, however if the candidate can be matched to some lexicon word then that word is the output. In the testing stage if the output word has any homophones they are ignored however the end system will output all homophones with the output word.

5.4 Format Table Encoding

The Format table contains all the different pairings of phonemes and graphemes in the dataset. The table contains 657 entries for the phonemes and all the part of speech tag combinations. The table is used to process the data, which is in human-readable format into numeric form, for Snob to process. Different arrangements of the Format Table result in different SCFG's produced by Snob and different ACP and ACPed results.

6 Results and Discussion

There were three systems compared during the testing phase of the study, with our system tested on two different datasets under two different alignment models. Table 3 is a comparison of our system with that of (Marchand and Damper, 2000), with access to different amounts of the dataset. Comparing our results with (Thomas et al., 1997) was difficult because they used the WER metric which is calculated differently to our ACP and ACPed and their results are omitted from the table. Our system performs less well than the other systems when only PTGC is considered, but does better than either of the other systems when the post-processing is included. Only one system, that of (Marchand and Damper, 2000), was suitable for leave-one-out testing but looks to be the most promising solution to the HLA problem if the low ACP can be further lowered by the factor seen in our system when the system has full access to the dataset. The two models we used to test our system reveal that A_N gives the lowest ACP under all types of access to the dataset and that including POS tags increases the ACP from 81.2% to 91.0% for that dataset (table 3).

Two of the systems studied performed tests on the effects of distortion to their systems. Tables 4 to 8 are the results of running the A_N system under various distortions. The distortions were performed by changing all occurrences of a phoneme in the dataset to a different phoneme and running the system again without retraining it. The values in the tables are the penalty costs for each distortion in the corresponding row and column. For example, in table 4 column 1, row 1 the value of the penalty is -0.73%, corresponding to a fall in the ACPed of 0.73%. The "Freq." row reports the frequency of the column element in the dataset.

Tables 5 and 4 rely on the PTGC component of the system to recover from errors and tables 6, 7 and 8 rely on the spellchecking software to recover. Distortion to the input of the PTGC does not degrade performance as much as to the post-processor under most conditions of distortion unless the distortion results in a phoneme which is found completely different environments to the original. Table 5 actually records a bonus, suggesting that some phonemes have interchangeable environments. Tables 6, 7 and 8 demonstrate that during post-processing recovering from errors is difficult, with an average decrease in performance of 30% (where table 1 expands the abbreviations).

References

José-Miguel Benedí and Joan-Andreu Sánchez. 2007. Fast Stochastic Context-Free Parsing: A Stochastic version of the Valiant Algorithm. In *IbPRIA (1)*, pages 80–88.

Stephen Clark and James. R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 104–111.

A. Levenstein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics-Doklandy*, volume 10.

Yannick Marchand and Robert I. Damper. 2000. A multistrategy approach to improving pronunciation by analogy. *Association for Computational Linguistics*, 26(2):195–219.

Jon D. Patrick. 1991. Snob: A program for discriminating between classes. Technical Report CS 91/151, Dept Computer Science, Monash University, Melbourne, Australia.

Panagiotis A. Rentzepopoulous and George K. Kokkinakis. 1996. Efficient multilingual phoneme-to-grapheme conversion based on hmm. *Computational Linguistics*, 22(3):351–376.

Ian Thomas, Ingrid Zukerman, Jonathan Oliver, David Albrecht, and Bhavani Raskutti. 1997. Lexical access for speech understanding using minimum message length encoding. In *In UAI97 Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 464–471. Morgan Kaufmann.

Chris S. Wallace and David M. Boulton. 1968. An information measure for classification. *Computer Journal*, 11(2):185–194.

Chris S. Wallace and D. L. Dowe. 2000. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, January.

Chris S. Wallace. 2005. *Statistical and Inductive Inference by Minimum Message Length*. Springer, Berlin, Germany.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 37-45

Automatic Acquisition of Training Data for Statistical Parsers

Susan Howlett and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{show0556,james}@it.usyd.edu.au

Abstract

The limitations of existing data sets for training parsers has led to a need for additional data. However, the cost of manually annotating the amount and range of data required is prohibitive. For a number of simple facts like those sought in Question Answering, we compile a corpus of sentences extracted from the Web that contain the fact keywords. We use a state-of-the-art parser to parse these sentences, constraining the analysis of the more complex sentences using information from the simpler sentences. This allows us to automatically create additional annotated sentences which we then use to augment our existing training data.

1 Introduction

Determining the syntactic structure of sentences is a necessary step in analysing the content of text for a range of language processing tasks such as Question Answering (Harabagiu et al., 2000) and Machine Translation (Melamed, 2004).

The structures that a parsing system assigns to sentences are governed by the grammar used. While some parsers make use of hand-crafted grammars, e.g. Riezler et al. (2002) and Briscoe et al. (2006), these typically cannot accommodate a wide variety of sentences and increasing coverage incurs significant development costs (Cahill et al., 2008). This has led to interest in automatic acquisition of grammars from raw text or automatically annotated data such as the Penn Treebank (Marcus et al., 1993).

Automatically-acquired grammars may be classified according to whether the learning algorithm used to estimate the language model is supervised or unsupervised. Supervised algorithms, e.g. Collins (1999) and Charniak (2000), require a large number of sentences already parsed according to the desired formalism, while unsupervised approaches, e.g. Bod (2006) and Seginer (2007), operate directly on raw text. While supervised approaches have generally proven more successful, the need for annotated training data is a major bottleneck.

Although the emergence of the Penn Treebank as a standard resource has been beneficial in parser development and evaluation, parsing performance drops when analysing text from domains other than that represented in the training data (Sekine, 1997; Gildea, 2001). In addition, there is evidence that language processing performance can still benefit from orders of magnitude more data (e.g. Banko and Brill (2001)). However, the cost of manually annotating the necessary amount of data is prohibitive.

We investigate a method of automatically creating annotated data to supplement existing training corpora. We constructed a list of facts based on factoid questions from the TREC 2004 Question Answering track (Voorhees, 2004) and the ISI Question Answer Typology (Hovy et al., 2002). For each of these facts, we extracted sentences from Web text that contained all the keywords of the fact. These sentences were then parsed using a state-of-the-art parser (Clark and Curran, 2007).

By assuming that the same grammatical relations always hold between the keywords, we use the analyses of simple sentences to constrain the analysis of more complex sentences expressing the same fact. The constrained parses then form additional training data for the parser. The results here show that parser performance has not been adversely affected; when the scale of the data collection is increased, we expect to see a corresponding increase in performance.

2 Background

The grammars used in parsing systems may be classified as either hand-crafted or automatically acquired. Hand-crafted grammars, e.g. Riezler et al. (2002) and Briscoe et al. (2006), have the advantage of not being dependent on any particular corpus, however extending them to unrestricted input is difficult as new rules and their interactions with existing rules are determined by hand, a process which can be prohibitively expensive (Cahill et al., 2008).

In contrast, the rules in acquired grammars are learned automatically from external resources, typically annotated corpora such as the Penn Treebank (Marcus et al., 1993). Here, the system automatically determines all the rules necessary to generate the structures exemplified in the corpus. Due to the automatic nature of this process, the rules determined may not be linguistically motivated, leading to a potentially noisy grammar.

Systems that learn their model of syntax from annotated corpora like the Penn Treebank are termed *supervised* systems; in *unsupervised* learning algorithms, possible sentence structures are determined directly from raw text. Bod (2006) presents an unsupervised parsing algorithm that out-performs a supervised, binarised PCFG and claims that this heralds the end of purely supervised parsing. However these results and others, e.g. Seginer (2007), are still well below the performance of state-of-the-art supervised parsers; Bod explicitly says that the PCFG used is not state-of-the-art, and that binarising the PCFG causes a drop in its performance.

By extracting the grammar from an annotated corpus, the bottleneck is shifted from writing rules to the creation of the corpus. The 1.1 million words of newswire text in the Penn Treebank has certainly been invaluable in the development and evaluation of English parsers, however for supervised parsing to improve further, more data is required, from a larger variety of texts.

To investigate the importance of training data in language processing tasks, Banko and Brill (2001) consider the problem of confusion set disambiguation, where the task is to determine which of a set of words (e.g. {*to*, *too*, *two*}) is correct in a particular context. Note that this is a problem for which large amounts of training data can be constructed extremely cheaply. They compare four different machine learning algorithms and find that with a training corpus of 1 billion words, there is little difference between the algorithms, and performance appears not to be asymptoting. This result suggests that for other language processing tasks, perhaps including parsing, performance can still be improved by large quantities of additional training data, and that the amount of data is more important than the particular learning algorithm used.

Quantity of data is not the only consideration, however. It has been shown that there is considerable variation in the syntactic structures used in different genres of text (Biber, 1993), suggesting that a parser trained solely on newswire text will show reduced performance when parsing other genres.

Evaluating parser performance on the Brown corpus, Sekine (1997) found that the best performance was achieved when the grammar was extracted from a corpus of the same domain as the test set, followed by one extracted from an equivalent size corpus containing texts from the same class (fiction or non-fiction), while parsers trained on a different class or different domain performed noticeably worse.

Comparing parsers trained on the Penn Treebank and Brown corpora, Gildea (2001) found that a small amount of training data from the same genre as the test set was more useful than a large amount of unmatched data, and that adding unmatched training data neither helped nor hindered performance.

To improve parsing performance on non-newswire genres, then, training data for these additional domains are necessary. The Penn Treebank project took 8 years and was an expensive exercise (Marcus et al., 1993), so larger and more varied annotated corpora are not likely to be forthcoming. Since unsupervised approaches still under-perform, it is necessary to explore how existing annotated data can be automatically extended, turning supervised into *semi-supervised* approaches.

3 Parsing with CCG

Combinatory Categorial Grammar (CCG; Steedman (2000)) is a mildly context sensitive, lexicalised grammar formalism, where each word in the sentence is assigned a *lexical category* (or *supertag*), either *atomic* or *complex*, which are then combined

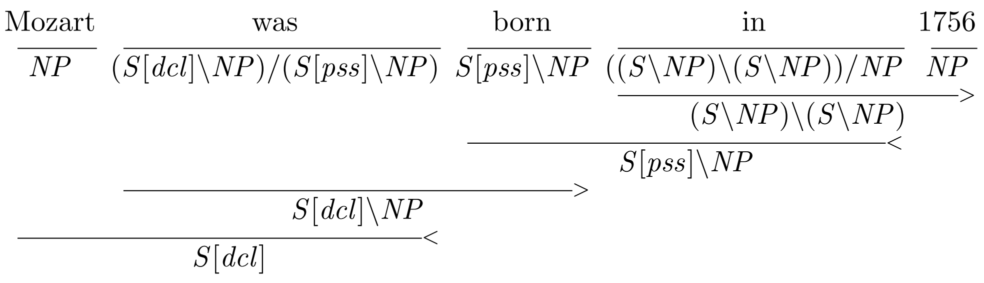


Figure 1: CCG derivation for the sentence *Mozart was born in 1756.*

using a small number of generic *combinatory rules*.

The possible atomic categories are N, NP, PP and S, representing nouns, noun phrases, prepositional phrases and clauses, respectively. They may carry additional features, e.g. $S[dcl]$ represents a declarative clause. Complex categories are binary structures consisting of two categories and a slash, in the form X/Y or $X\backslash Y$. These represent constituents that when combined with a constituent with category Y (*argument* category) form a constituent with category X (*result* category). The forward and backward slashes indicate that the Y is to be found to the right and left of X, respectively. Examples of complex categories are NP/N (determiner) and $(S\backslash NP)/NP$ (transitive verb).

The basic combinatory rules are *forward* and *backward application*, where complex categories are combined directly with their arguments to form the result. That is, $X/Y\ Y \Rightarrow X$ and $Y\ X\backslash Y \Rightarrow X$. Instances of these rules are marked by the symbols $>$ and $<$ respectively. A derivation illustrating these rules is given in Figure 1. Here, the word *in* with category $((S\backslash NP)\backslash(S\backslash NP))/NP$ combines with *1756*, an NP, using forward application to produce the constituent *in 1756* with category $(S\backslash NP)\backslash(S\backslash NP)$.

In addition, CCG has a number of rules such as *forward composition* ($>\mathbf{B}$) ($X/Y\ Y/Z \Rightarrow X/Z$) and type-raising capabilities (e.g. $X \Rightarrow T/(T\backslash X)$, represented by $>\mathbf{T}$) that increase the grammar's expressive power from context free to mildly context sensitive. These allow more complex analyses where, instead of the verb taking its subject NP as an argument, the NP takes the verb as an argument. These additional rules mean that the sentence in Figure 1 may have alternative CCG analyses, for example the one shown in Figure 2. This *spurious ambiguity* has been shown not to pose a problem to efficient parsing with CCG (Clark and Curran, 2007).

As each of the combinatory rules is applied, predicate-argument dependencies are created, of the form $\langle h_f, f, s, h_a, l\rangle$, where f is the category that governs the dependency, h_f is the word carrying the category f, s specifies which dependency of f is being filled, h_a is the head word of the argument, and l indicates whether the dependency is local or long-range. For example, in Figure 1, the word *in* results in the creation of two dependencies, both local:

$$\langle \textit{in},\ ((S\backslash NP)\backslash(S_1\backslash NP))/NP_2,\ 1,\ \textit{born},\ -\rangle$$
$$\langle \textit{in},\ ((S\backslash NP)\backslash(S_1\backslash NP))/NP_2,\ 2,\ \textit{1756},\ -\rangle$$

The values for s refer back to the subscripts given on the category f. Although the spurious derivations such as those in Figure 2 have a different shape, they result in the creation of the same set of dependencies, and are therefore equally valid analyses.

The predicate-argument dependencies to be formed are governed by variables associated with the categories nested within the category f. For example, *in* is more fully represented by the category $(((S_Y\backslash NP_Z)_Y\backslash(S_{Y,1}\backslash NP_Z)_Y)_X/NP_{W,2})_X$. These variables represent the heads of the corresponding constituents; the variable X always refers to the word bearing the lexical category, in this case *in*.

During the derivation, dependencies form between the word filling variable X and the word filling the variable directly associated with the dependency. In this example, the first dependency forms between *in* and the word filling variable Y, and the second forms between *in* and the word filling W. When *in* combines with *1756* through forward application, *1756* is identified with the argument $NP_{W,2}$, causing W to be filled by *1756*, and so a dependency forms between *in* and *1756*.

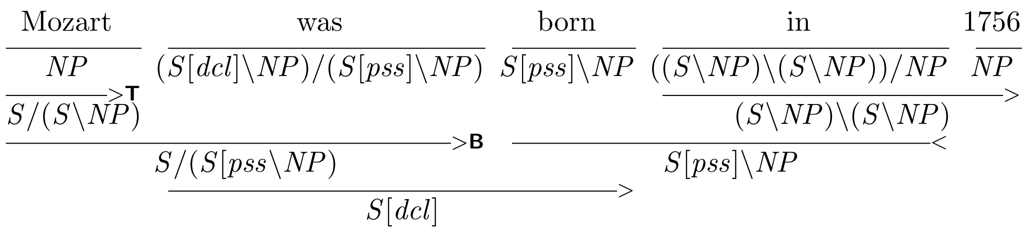


Figure 2: Alternative derivation for the sentence in Figure 1, illustrating CCG's spurious ambiguity.

4 The C&C Parser

The parser we use (Clark and Curran, 2007) is a state-of-the-art parser based on CCG. The grammar is automatically extracted from CCGbank, a conversion of the Penn Treebank into CCG (Hockenmaier, 2003). As the training data is ultimately from the Penn Treebank, the grammar is subject to the same difficulties mentioned in Section 2.

The two main components of this parser are the supertagger, responsible for assigning each word its possible CCG lexical categories, and the parser, which combines the lexical categories to form the full derivation, using the CKY chart parsing algorithm as described in Steedman (2000). The supertagger and parser are tightly integrated so that, for example, if a spanning analysis is not found, more lexical categories can be requested.

The third component, the decoder, chooses the most probable of the spanning analyses found by the parser, according to the statistical model used. Clark and Curran (2007) discuss three separate models, one of which involves finding the most probable derivation directly, while the other two involve optimising the dependency structure returned. Here we use the first model, called the normal-form model.

Clark and Curran (2007) evaluate their parser in two different ways. Their main method of evaluation is to compare the parser's predicate-argument dependency output against the dependencies in CCGbank. They calculate labelled and unlabelled precision, recall and F-score for the CCG dependencies plus category accuracy, the percentage of words assigned the correct lexical category.

This method of evaluation, however, does not allow easy comparison with non-CCG systems. Evaluating a CCG parser using the traditional metrics of precision, recall and F-score over Penn Treebank bracketings is problematic since CCG derivations, being binary-branching, can have a very different shape from the trees found in the Penn Treebank. This is particularly true of the spurious derivations, which will be heavily penalised even though they are correct analyses that lead to the correct predicate-argument dependencies.

To address this problem, Clark and Curran (2007) also evaluate their parser against the Briscoe and Carroll (2006) re-annotation of the PARC Dependency Bank (DepBank; King et al. (2003)), which contains 700 sentences from section 23 of the Penn Treebank, represented in the form of grammatical relations, e.g. `ncsubj` (non-clausal subject) and `dobj` (direct object). To do this, they convert the predicate-argument dependency output of the parser into grammatical relations, a non-trivial, many-to-many mapping. The conversion process puts the C&C parser at a disadvantage, however the its performance still rivals that of the RASP parser (Briscoe et al., 2006) that returns DepBank grammatical relations natively. Figure 3 illustrates the grammatical relations obtained from the analysis in Figure 1.

5 Getting Past the Bottleneck

Although some work has aimed at reducing the cost of training the C&C parser (Clark and Curran, 2006), the question remains of whether annotated training data can be obtained for free.

In their first entry to the TREC Question-Answering task, Brill et al. (2001) reasoned that while finding the correct answer in the given corpus may sometimes require sophisticated linguistic or logical processing, a preliminary answer found on the Web can help to identify the final answer in the corpus. They argue that the sheer size of the Web means that an answer can often be found using simple or shallow processing. We exploit the same idea of the redundancy of information on the Web here.

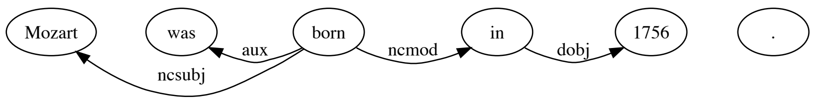


Figure 3: Grammatical relations for *Mozart was born in 1756.*

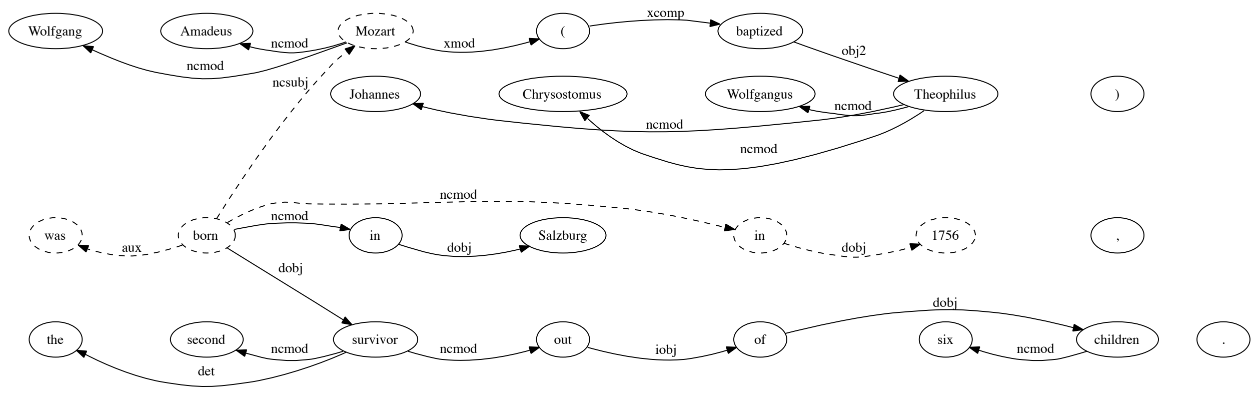


Figure 4: Desired analysis for the more complex sentence, with the analysis in Figure 3 indicated by dashed lines.

A parsing model trained on existing data can already confidently parse a simple sentence such as *Mozart was born in 1756* (Figure 3). Given the size of the Web, many such sentences should be easily found. A longer sentence, such as *Wolfgang Amadeus Mozart (baptized Johannes Chrysostomus Wolfgangus Theophilus) was born in Salzburg in 1756, the second survivor out of six children*, is more complex and thus contains more opportunities for the parser to produce an incorrect analysis. However, this more complex sentence contains the same grammatical relations between the same words as in the simple sentence (Figure 4).

Similar to the *one sense per collocation* constraint (Yarowsky, 1993), we assume that any sentence containing the words *Mozart*, *born* and *1756* will contain the same relationships between these words. We hypothesise that by constraining the parser to output an analysis consistent with these relationships, the correct analysis of the complex sentences can be found without manual intervention. These complex sentences can then be used as additional training data, allowing the parser to learn the general pattern of the sentence. For this process, we use grammatical relations rather than CCG dependencies as the former generalise across the latter and are therefore more transferable between sentences.

6 Method

Our procedure may be outlined as follows:

1. Manually select facts and identify keywords.
2. Collect sentences from HTML documents containing all keywords of a given fact.
3. Manually identify grammatical relations connecting fact keywords.
4. Parse all sentences for the fact, using these grammatical relations as constraints.
5. Add successful parses to training corpus.

Further detail is given below. Section 6.1 covers items 1 and 2, 6.2 item 3 and 6.3 items 4 and 5.

6.1 Sentence collection

First, we compiled a list of 43 facts based on factoid questions from the TREC 2004 Question Answering track (Voorhees, 2004) and the ISI Question Answer Typology (Hovy et al., 2002). In order for our hypothesis to hold, it is necessary for the facts used to refer to relatively unambiguous entities. For each fact, we identified a set of keywords and submitted these as queries through the Google SOAP Search API. The query was restricted to files with `.html` or `.htm` extensions to simplify document processing. Each unique page returned was saved.

Prior to splitting the documents into sentences, HTML tags were stripped from the text and HTML character entities replaced with their character equivalents. In order to incorporate some HTML markup information in the sentence boundary identification process, each page was split into a number of chunks by dividing at certain manually-identified HTML tags, such as heading and paragraph markers, which are unlikely to occur mid-sentence. Each of these chunks was then passed through the sentence boundary identifier available in NLTK v.0.9.3[1] (Kiss and Strunk, 2006). Ideally, the process would use a boundary identifier trained on HTML text including markup, to avoid these heuristic divisions.

To further simplify processing, sentences were discarded if they contained characters other than standard ASCII alphanumeric characters or a small number of additional punctuation characters. We also regarded as noisy and discarded sentences containing whitespace-delimited tokens that contained fewer alphanumeric characters than other characters.

Sentences were then tokenised using a tokeniser developed the C&C parser for the TREC competition (Bos et al., 2007). Sentences longer than 40 tokens were discarded as this might indicate noise or an error in sentence identification. Finally, sentences were only kept if each fact keyword appeared exactly once in the tokenised sentence, to avoid having to disambiguate between repetitions of keywords.

At the conclusion of this process, each fact had between 0 and 299 associated sentences, for a total of 3472 sentences. 10 facts produced less than 10 sentences; the average yield of the remaining 33 facts was just over 100 sentences each. This processing does result in the loss of a considerable number of sentences, however we believe the quality and level of noise in the sentences to be a more important consideration than the quantity, since there are still many facts that could yet be used.

6.2 Constraint identification

For each of the 33 facts that yielded more than 10 sentences, we identified a set of grammatical relations that connects the keywords in simple sentences. These sets contained between two and six grammatical relations; for example the relations for the Mozart fact were `(ncsubj born Mozart)`, `(ncmod born in)` and `(dobj in 1756)`. Since we assume that the keywords will be related by the same grammatical relations in all sentences (see Section 5), we use these grammatical relations as constraints on the analyses of all sentences for the corresponding fact. Future work will automate this constraint identification process.

To avoid the need to disambiguate between instances of particular words, when constraints are applied each word must be identified in the sentence uniquely. Although sentences that contained repetitions of fact keywords were discarded during the collection stage, the constraints identified in this stage of the process may incorporate additional words. In the Mozart fact, the constraints contain the word *in* in addition to the keywords *Mozart*, *born* and *1756*. The sentence in Figure 4, for example, contains two instances of *in*, so this sentence is discarded at this point, along with other sentences that contain multiple copies of a constraint word.

6.3 Parsing with constraints

Having identified the grammatical relations to use as constraints, the sentences for the corresponding fact are parsed with the constraints enforced using the process described below. Sentences from two of the 33 facts were discarded at this point because their constraints included grammatical relations of type `conj`, which cannot be enforced by this method.

As described in Section 3, dependencies are formed between variables on lexical categories. For example, a dependency is created between the words *in* and *1756* in Figure 1 by filling a variable W in the category of *in*. To force a particular dependency to be formed, we determine which of the two words will bear the category that lists the dependency; this word we call the *head*. We identify the variable associated with the desired dependency and pre-fill it with the second word. The parser's own unification processes then ensure that the constraint is satisfied.

Since the constraints are expressed in terms of grammatical relations and not CCG dependencies, each constraint must first be mapped back to a number of possible dependencies. First, the head word may be assigned several possible categories by the supertagger. In this case, if any category does not license the desired dependency, it is removed from

[1]http://nltk.sourceforge.net

			Normal			Constrained	
Keywords	# Pages	# Sents	# Parse	# Fail	# Discarded	# Parse	# Fail
Armstrong landed Moon 1969	805	20	20	0	14	0	6
CNN Cable News Network	286	62	62	0	41	8	13
Canberra capital Australia	601	78	77	1	64	0	14
Columbus discovered America	683	299	291	8	0	174	125
Martin Luther King Jr assassinated 1968	675	6	6	0	–	–	–
Mozart born 1756	734	161	160	1	93	30	38
hydrogen lightest element	582	85	83	2	51	9	25
Total (all 43 facts)	24934	**3472**	3421	51	–	–	–
Total (31 facts for which constraints successfully applied)	19019	**3237**	3190	47	1661	**662**	914

Table 1: Number of sentences acquired for a selection of facts. Figures are given for the number of sentences parsed and failed by the normal (unconstrained) parser as an indication of parser coverage on Web sentences.

the list; all others have a variable filled. In this way, no matter which category is chosen, the constraint is enforced. Secondly, one grammatical relation may map to some dependencies governed by the first word and some governed by the second. That is, in some constraints, either word may operate as the head. To account for this, we parse the sentence several times, with different combinations of head choices, and keep the highest probability analysis.

When parsing with constraints, not all sentences can be successfully parsed. Those where a spanning analysis consistent with the constraints cannot be found are discarded; the remainder are added to the training corpus. From the 31 facts still represented at this stage, we obtained 662 sentences.

7 Results

Table 1 traces the quantity of data throughout the process described in Section 6, for a sample of the facts used. It also gives totals for all 43 facts and for the 31 facts represented at completion.

The first column lists the fact keywords. We used a range of fact types, including abbreviations (CNN stands for Cable News Network) and identities (Canberra is the capital of Australia, hydrogen is the lightest element) as well as actions (Armstrong landed on the Moon in 1969) and passive structures (Martin Luther King, Jr. was assassinated in 1969). This last example is one where fewer than 10 sentences were collected, most likely due to the large number of keywords.

The # Pages column in Table 1 shows the number of unique Web pages saved for each fact, while # Sents indicates the number of unique sentences containing all fact keywords, after tokenisation. The next two figures show the coverage of the baseline parser on the Web sentences. The # Discarded column indicates the number of these sentences that were discarded during the constraining process due to constraint words being absent or appearing more than once. The final two columns show how many of the remaining sentences the parser found a spanning analysis for that was consistent with the constraints.

It is clear that there is considerable variation in the number of sentences discarded during the constraining process. For some facts, such as the Columbus fact, the keywords form a chain of grammatical relations with no additional words necessary. In such cases, no sentences were discarded since any sentences that reached this point already met the unique keyword restriction.

The main reason for sentences being discarded during constraining is that some relationships are expressed in a number of different ways. Prepositions and punctuation are frequently responsible for this. For example, the date of the Moon landing can be expressed as both *in 1969* and *on July 20, 1969*, while the expansion of the abbreviation CNN may be expressed as *CNN: Cable News Network*, *CNN (Cable News Network)* and indeed *Cable News Net-*

Model	LP	LR	LF	LF (POS)	SENT ACC	UP	UR	UF	CAT ACC	COV
Baseline	85.53	84.71	85.12	83.38	32.14	92.37	91.49	91.93	93.05	99.06
New	85.64	84.77	85.21	83.54	32.03	92.41	91.47	91.94	93.08	99.06

Table 2: Performance of the baseline (Clark and Curran (2007) normal-form model) and new parser models on CCGbank section 23. Most results use gold standard POS tags; LF (POS) uses automatically assigned tags.

work (CNN). By selecting only one set of constraints, only one option can be used; sentences involving other formats are therefore discarded. To overcome these difficulties, it would be necessary to allow several different constraint chains to be applied to each sentence, taking the highest probability parse from all resulting analyses. This could also be used to overcome the word-uniqueness restriction.

To evaluate the effectiveness of the 662 additional sentences, we trained two models for the C&C parser, a baseline using the parser's standard training data, sections 02–21 of CCGbank, and a second model using CCGbank plus the extra data. For details of the training process, see Clark and Curran (2007). Following Clark and Curran (2007), we evaluate the performance of the two parsing models by calculating labelled and unlabelled precision, recall and F-score over CCGbank dependencies, plus coverage, category accuracy (percentage of words with the correct lexical category) and sentence accuracy (percentage of sentences where all dependencies are correct).

The difficulty we face here is that the additional data is Web text, while the evaluation data is still newswire text. Since genre effects are important, we cannot expect to see a large difference in results in this evaluation. Future work will compare the two models on a corpus of Web sentences collected according to the same procedure, using different facts.

Table 2 summarises the results of our evaluation. The figures for the performance of the baseline system are the latest for the parser, and slightly higher than those given in Clark and Curran (2007). It is clear that the results for the two systems are very similar and that adding 662 sentences, increasing the amount of data by approximately 1.7%, has had a very small effect. However, now that the automated procedure has been developed, we are in a position to substantially increase this amount of data without requiring manual annotation.

8 Conclusion

We have described a procedure for automatically annotating sentences from Web text for use as training data for a statistical parser. We assume that if several sentences contain the same set of relatively unambiguous keywords, for example *Mozart*, *born* and *1756*, then those words will be connected by the same chain of grammatical relations in all sentences. On this basis, we constrain a state-of-the-art parser to produce analyses for these sentences that contain this chain of relations. The constrained analyses are then used as additional training data for the parser.

Aside from the initial identification of facts, the only manual step of this process is the the choice of constraints. The automation of this step will involve the identification of reliable sentences, most likely short sentences consisting of only a single clause, from which the relation chain can be extracted. The chain will need to be supported by a number of such sentences in order to be accepted. This is the step that truly exploits the redundancy of information on the Web, as advocated by Brill et al. (2001).

Once the choice of constraints has been automated, we will have a fully automatic procedure for creating additional annotated training data for a statistical parser. Our initial results show that the training data acquired in this manner can be used to augment existing training corpora while still maintaining high parser performance. We are now in a position to increase the scale of our data collection to determine how performance is affected by a training corpus that has been significantly increased in size.

Acknowledgements

We would like to thank the anonymous reviewers and the Language Technology Research Group at the University of Sydney for their comments. The first author is supported by a University of Sydney Honours scholarship.

References

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Assocation for Computational Linguistics*, pages 26–33.

Douglas Biber. 1993. Using register-diversified corpora for general language studies. *Computational Linguistics*, 19(2):219–241, June.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872.

Johan Bos, James R. Curran, and Edoardo Guzetti. 2007. The Pronto QA system at TREC-2007. In *Proceedings of the Sixteenth Text REtreival Conference*.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Y. Ng. 2001. Data-intensive question answering. In *Proceedings of the Tenth Text REtreival Conference*, pages 393–400, November.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 41–48, July.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia, July.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124, March.

Eugene Charniak. 2000. A maxmium entropy inspired parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 144–151, June.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, December.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Milhalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtreival Conference*.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Eduard Hovy, Ulf Hermjakob, and Deepak Ravichandran. 2002. A question/answer typology with surface text patterns. In *Proceedings of the DARPA Human Language Technology conference*, pages 247–251.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora*, pages 1–8.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 653–660.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional gramar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 384–391, Prague, Czech Republic, June.

Satoshi Sekine. 1997. The domain dependence of parsing. In *Proceedings of the fifth conference on Applied Natural Language Processing*, pages 96–102.

Mark Steedman. 2000. *The Syntactic Process*. Language, Speech, and Communication series. The MIT Press, Cambridge, MA.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Question Answering track. In *Proceedings of the Thirteenth Text REtreival Conference*.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 46-54

Writing Support for Controlled Natural Languages

Tobias Kuhn
Department of Informatics
University of Zürich
Switzerland
tkuhn@ifi.uzh.ch

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

Abstract

In this paper we present interface techniques that support the writing process of machine-oriented controlled natural languages which are well-defined and tractable fragments of English that can be translated unambiguously into a formal target language. Since these languages have grammatical and lexical restrictions, it is important to provide a text editor that assists the writing process by using lookahead information derived from the grammar. We will discuss the requirements to such a lookahead text editor and introduce the semantic wiki AceWiki as an application where this technology plays an important role. We investigate two different approaches how lookahead information can be generated dynamically while a text is written and compare the runtimes and practicality of these approaches in detail.

1 Introduction

Natural language interfaces have been a popular area of research in the 70's and 80's, in particular natural language interfaces to databases received a lot of attention and some of these interfaces found their way into commercial applications (Copestake and Jones, 1990; Androutsopoulos et al., 1995). However, most of the early research prototypes focused on the exploration of specific NLP techniques and formalisms and were not very robust and therefore not very successful – and research drifted away from this topic in the 90's although there is still no agreed consensus theory on how to build these text-based interfaces.

The need for text-based natural language interfaces has recently gained again momentum since more and more non-specialists are required to access databases and maintain knowledge systems in Semantic Web applications through their web browsers and portable devices (Cimiano et al., 2008). One advantage of natural language interfaces is that the user is not required to learn a formal language to communicate with the system. But the use of natural language as an interface style is not without problems since full natural language is highly ambiguous and context-dependent. In practice, natural language interfaces can only understand a restricted subset of written (or spoken) input, and they have often only limited capabilities to make their coverage transparent to the user (Chintaphally et al., 2007). If the system is unable to understand a sentence and does not provide accurate feedback, then there is the risk that the user makes potentially incorrect assumptions about other sentences that the system actually can understand.

An obvious solution to this problem is to use a set of training sessions that teach the user what the system can and cannot understand. But this solution has at least two drawbacks: (a) untrained users cannot immediately use the system; and (b) infrequent users might not remember what they learned about the system's capabilities (Tennant et al., 1983a).

A better solution is to build natural language interfaces that directly support the interaction between the user and the system and that are able to enforce the restrictions of the language and guide the writing process in an unobtrusive way.

Menu-based natural language interfaces have been suggested to overcome many problems of conventional text-based natural language interfaces. These menu-based interfaces can be generated automatically from the description of a database or from a predefined ontology (Tennant et al., 1983b). These interfaces are very explicit about their coverage since they provide a set of (cascaded) menus that contain the admissible natural language expressions and the user has to select only from a set of choices. Therefore the failure rate is very low, in addition there are no spelling errors since no typing is required. Most menu-based and related WYSIWYM (= What You See Is What You Mean) approaches assume an existing underlying formal representation (Power et al., 1998). WYSIWYM is a natural language generation approach based on conceptual authoring where the system generates feedback from a formal representation. Instead of manipulating the formal representation, the user edits the feedback text via a set of well-defined editing operations (Hallett et al., 2007).

Although menu-based natural language interfaces can be applied to large domains, there exist applications where menu-searching becomes cumbersome (Hielkema et al., 2008) and more importantly there exist many applications where the formal representation does not exist in advance and where the construction of this formal representation is the actual task. This is the setting where controlled natural languages can make an important contribution. Machine-oriented controlled natural languages can be used as high-level interface languages for creating these formal representations, and we can try to support the writing process of these controlled natural languages in an optimal way.

In the following, we will illustrate how controlled natural languages can be applied in a semantic wiki context where a formal representation needs to be derived from a text that is human-readable as well as machine-processable, and then we will show how these techniques can be implemented in a logic programming framework.

2 Controlled Natural Languages (CNL)

Over the last decade or so, a number of machine-oriented controlled natural languages have been designed and used for specification purposes, knowledge acquisition and knowledge representation, and as interface languages to the Semantic Web – among them Attempto Controlled English (Fuchs et al., 1998; Fuchs et al., 2008), PENG Processable English (Schwitter, 2002; Schwitter et al., 2008), Common Logic Controlled English (Sowa, 2004), and Boeing's Computer-Processable Language (Clark et al., 2005; Clark et al., 2007). These machine-oriented controlled natural languages are engineered subsets of a natural language with explicit constraints on grammar, lexicon, and style. These constraints usually have the form of construction and interpretation rules and help to reduce both ambiguity and complexity of full natural language.

The PENG system, for example, provides text- and menu-based writing support that takes the burden of learning and remembering the constraints of the language from the user and generates a paraphrase that clarifies the interpretation for each sentence that the user enters. The text editor of the PENG system dynamically enforces the grammatical restrictions of the controlled natural language via lookahead information while a text is written. For each word form that the user enters, the user is given a list of choices of how to continue the current sentence. These choices are implemented as hypertext links as well as a cascade of menus that are incrementally updated. The syntactic restrictions ensure that the text follows the rules of the controlled natural language so that the text is syntactically correct and can be translated unambiguously into the formal target language (Schwitter et al., 2003; Schwitter et al., 2008).

3 Semantic Wikis

Semantic wikis combine the philosophy of wikis (i.e. quick and easy editing of textual content in a collaborative way over the Web) with the concepts and techniques of the Semantic Web (i.e. giving information well-defined meaning in order to enable computers and people to work in cooperation). The goal is to manage formal representations within a wiki environment.

There exist many different semantic wiki systems. Semantic MediaWiki (Krötzsch et al., 2007), IkeWiki (Schaffert, 2006), and OntoWiki

(Auer et al., 2006) belong to the most mature existing semantic wiki engines. Unfortunately, **none** of the existing semantic wikis supports expressive ontology languages in a general way. For example, none of them allows the users to define general concept inclusion axioms, e.g.:

- Every country that borders no sea is a landlocked country.

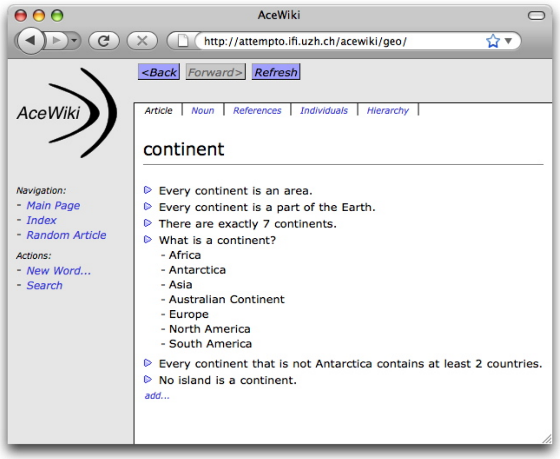
Furthermore, most of the existing semantic wikis fail to hide the technical aspects, are hard to understand for people who are not familiar with the technical terms, and do not provide any writing support.

4 AceWiki – a CNL-based Wiki

AceWiki (Kuhn, 2008a; Kuhn, 2008b) is a semantic wiki that tries to solve those problems by using the controlled natural language Attempto Controlled English (ACE) to represent the formal content.[1] The use of the language ACE allows us to provide a high degree of expressivity and to represent the formal content in a natural way. Easy creation and modification of the content is enabled by a special text editor that supports the writing process.

AceWiki is implemented in Java. Making use of the Echo Web Framework[2], it provides a rich AJAX-based web interface. Figure 1 shows a screenshot of an exemplary AceWiki instance containing information about a geographical domain. The ACE parser is used to translate the ACE sentences into first-order logic and, if possible, into the ontology language OWL (Motik et al., 2008). In the background, the OWL reasoner Pellet[3] is used to ensure that the ontology (consisting of the sentences that are OWL-compliant) is always consistent. The reasoner is also used to infer class memberships and hierarchies and to answer questions.

Following the wiki philosophy, the users should be able to change the content of the wiki quickly and easily. For this reason, it is important that the users are supported by an intelligent text editor that helps to construct valid sentences. AceWiki provides a predictive text editor that is



continent

- Every continent is an area.
- Every continent is a part of the Earth.
- There are exactly 7 continents.
- What is a continent?
  - Africa
  - Antarctica
  - Asia
  - Australian Continent
  - Europe
  - North America
  - South America
- Every continent that is not Antarctica contains at least 2 countries.
- No island is a continent.

Figure 1: A screenshot of the web interface of AceWiki showing the wiki article for the class *continent*.

able to look ahead and to show possible words to continue the sentence. Figure 2 shows a screenshot of this editor. A chart parser is used for the generation of the lookahead information.

This predictive editor enables the creation of sentences by clicking consecutively on the desired words. It ensures that only 100% syntactically correct sentences are created. This enables novice users to create ACE sentences without the need to read the ACE manuals first. Alternatively, the predictive editor allows more experienced users to type a sentence or a part of it into a text field. The two modes of sentence construction (clicking versus typing) can be combined and the users can switch from one mode to the other at any time. Content words that are not yet known can be added on-the-fly when writing a sentence using the built-in lexical editor.

A usability experiment (Kuhn, 2008a) showed that people with no background in formal methods are able to work with AceWiki and its predictive editor. The participants — without receiving instruction on how to use the interface — were asked to add general and verifiable knowledge to AceWiki. About 80% of the resulting sentences were semantically correct and sensible statements (in respect of the real world). More than 60% of those correct sentences were complex in the sense that they contained an implication or a negation.

[1]`http://attempto.ifi.uzh.ch/acewiki/`
[2]`http://echo.nextapp.com/site/`
[3]`http://pellet.owldl.org/`

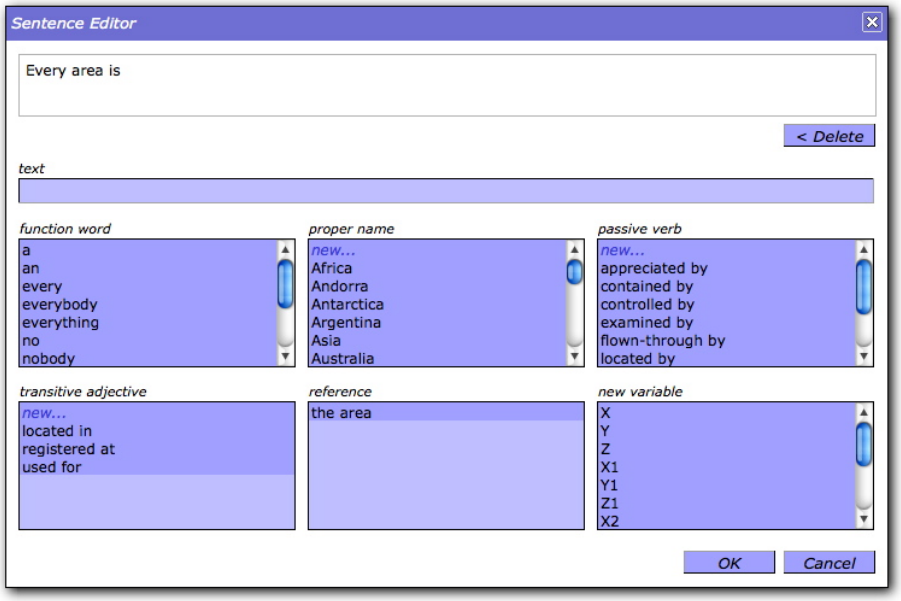


Figure 2: A screenshot of the predictive editor of AceWiki. The partial sentence *Every area is* has already been entered and now the editor shows all possibilities to continue the sentence. The possible words are arranged by their type in different menu boxes.

5 AceWiki Grammar

AceWiki uses a subset of ACE. The grammar contains about 80 grammar rules and is implemented in a declarative way in Prolog. These rules are defined in a special grammar notation using feature structures, for example:

```
verbphrase(pl:PL,neg:Neg,that:T) =>
  verb(verb:full,pl:PL,neg:Neg),
  nounphrase(verb:full,that:T).
```

This format can be transformed into Java code (which is used by the AceWiki chart parser) and can also be exported as a Prolog DCG grammar (Pereira and Shieber, 1987).

The AceWiki language consists of about 30 function words (e.g. *a*, *every*, *if*, *then*, *and*, *not*, *is*, *that*) and five extensible types of content words (some of which have several word forms): proper names (e.g. *Europe*), nouns (e.g. *city*), transitive verbs (e.g. *contains*), *of*-constructions (e.g. *part of*), and transitive adjectives (e.g. *located in*).

The grammar covers a considerable part of English: singular and plural noun phrases, active and passive voice, negation, relative phrases, conjunctions/disjunctions (of sentences, verb phrases, and relative phrases), existential and universal quantifiers, questions, and anaphoric references. Below are a few examples of declarative sentences:

- Switzerland is located in Europe and borders exactly 5 countries.
- Every country that borders no sea is a landlocked country and every landlocked country is a country that borders no sea.
- If X borders Y then Y borders X.

The semantic expressivity of the AceWiki language is a subset of the expressivity of first-order logic. The mapping from ACE to OWL that is applied by the ACE parser covers all of OWL 2 except data properties and some very complex class descriptions (Kaljurand, 2007). Apart from those exceptions, the AceWiki language is more expressive than OWL. The examples shown above are OWL-compliant, but we can write sentences that go beyond the semantic expressivity of OWL, for example[4]:

- No country borders every country.
- If Berlin is a capital then Germany is not an unstable country.
- If a country contains an area and does not control the area then the country is an unstable country.

[4](Kaljurand, 2007) explains how OWL-compliant ACE sentences can be distinguished from ACE sentences that have no OWL representation.

At the moment, AceWiki supports only simple questions containing exactly one *wh*-word (i.e. *what*, *who*, or *which*). Such questions can be mapped to OWL class descriptions[5]. The answers to such questions can be calculated by determining the individuals of the class description. Here are three typical questions expressed in AceWiki:

- Which continents contain more than 10 countries?
- Switzerland borders which country that borders Spain?
- Which rivers flow through a country that borders Spain?

In summary, the AceWiki grammar describes a relatively large subset of English which exceeds the expressivity of OWL.

6 Implementing Writing Support

As we have seen, the AceWiki grammar is written in a format that can be read as a DCG and that is used by the AceWiki chart parser in order to generate lookahead information and logical formulas. In this section, we will focus on a specific aspect of the writing support and illustrate how syntactic lookahead information can be harvested from the DCG in a direct way and in an indirect way through a chart parser (as AceWiki does). In order to make a direct comparison of the two approaches, we use Prolog in both cases.

6.1 Harvesting the DCG

In order to harvest lookahead information directly from the DCG (for which Prolog supplies by default a top-down, left-to-right, backtrack parsing algorithm), we add a dummy token after each new word that the user enters and then parse the entire input string from the beginning. This looks like a very costly approach but the advantage of this approach is that it can be implemented very easily (and our empirical results show that this approach is still fast enough for practical applications). Furthermore, this approach prevents the parser from getting into infinite loops when processing recursive grammar rules. Let us assume that the user is planning to add the following sentence to the wiki:

- Switzerland is an alpine country.

[5]These queries are sometimes called "DL Queries".

and that she has just appended the indefinite article *an* to the string *Switzerland is*. Before this entire input string is parsed by the DCG parser, the dummy token `'$dummy$'` is added to the end of the string:

['Switzerland',is,an,'$dummy$']

This dummy token will be used – as we will see below – to trigger a Prolog rule (`word_form/4`) that processes the lookahead information. Before we describe this, we need to understand how preterminal grammar rules look like in the AceWiki grammar; here is a (simplified) example:

```
'NOUN' -->
  { lexicon(cat:'NOUN',wf:WF) },
  word_form('NOUN',WF).
```

The term in curly brackets is a Prolog expression. The default Prolog DCG preprocessor takes this DCG rule and translates it into a pure Prolog clause, adding two additional arguments for the input and output string. Note that the variable `WF` stands for an entire word form that can either be atomic or compound. This word form is represented as a list of tokens in the lexicon, for example (again simplified here):

```
lexicon(cat:'NOUN',
        wf:[alpine,country]).
```

This lexicon lookup (`lexicon/2`) returns the entire list of tokens and sends it to the rule `word_form/4` that processes these tokens depending on the input string:

```
word_form(Cat,[T1|Ts],['$dummy$'],_) :-
  update_lah_info(lookahead(Cat,T1)),
  fail.

word_form(Cat,[T1|Ts],[T1|S1],S2) :-
  word_form(Cat,Ts,S1,S2).
```

At this point, the remaining input string contains only the dummy token since all other words of this string have already been processed. This dummy token is processed by the first of the two grammar rules (`word_form/4`) that takes the first token (`T1`) of the compound word and adds it together with the category (`Cat`) to the lookahead information. Once the lookahead information is updated, the rule fails and additional lookahead information is collected via backtracking. The text editor will display the first token (`alpine`) of the compound word (`alpine country`) together with other lookahead information. If the user selects alpine as the subsequent input, the dummy token is added again to the new input string but this time the second of the grammar rules (word_form/4) is triggered. This rule removes alpine from the input string before the dummy token is detected by the first rule that adds the next token of the compound (country) to the lookahead information. In order to avoid duplicates, the predicate update_lah_info/1 first checks if the lookahead information already exists and only asserts this information if it is new:

```
update_lah_info(LAHInfo) :-
  (
    call(LAHInfo), !
  ;
    assert(LAHInfo)
  ).
```

Instead of atomic and compound word forms, we can also display syntactic categories for entire groups of words using the same technique.

6.2 Harvesting the Chart

As we have seen in the last section, the DCG is used to parse each sentence from scratch to process the lookahead information for a new word. Apart from that the DCG creates many partial structures and destroys them while backtracking. In order to avoid unnecessary repetition of work, the DCG can be processed with a (top-down) chart parser. A chart parser stores well-formed constituents and partial constituents in a table (= chart) consisting of a series of numbered vertices that are linked by edges (see Kay (1980) or Gazdar and Mellish (1989) for an introduction). Our chart parser is an extension of (Gazdar and Mellish, 1989) and represents edges in a similar way as a predicate with five arguments:

edge(V1,V2,LHS,RHSFound,RHSToFind)

The first two arguments state that there exists an edge between vertex V1 and vertex V2. The next three arguments represent a grammar rule and indicate to what extent the parser was able to apply this grammar rule to the constituent found between the two vertices. Here LHS stands for a category on the left-hand side of a grammar rule, RHSFound for a sequence of categories that has been found on the right-hand side of the grammar rule, and RHSToFind for a sequence of remaining categories on the right-hand side. The edges in the chart are either *inactive* or *active* edges. An inactive edge is an edge where the list RHSToFind is empty ([]) and represents a confirmed hypothesis. All other edges are active and represent unconfirmed hypotheses.

The fundamental rule of chart parsing says that if an inactive edge meets an active edge of the corresponding category, then a new edge can be added to the chart that spans both the inactive and the active edges. In order to apply the fundamental rule, the chart needs to contain at least one active and one inactive edge. That means we have to initialise the chart first for each new sentence. This initialisation process will allow us to add a number of active edges to the chart and to extract the initial lookahead information from the chart. This information will show the user how to start a sentence. Once the user selects or types the first word, new inactive edges are triggered that are required by the fundamental rule to start the chart parsing process.

Before we can process the DCG with the chart parser, we transform it into a slightly different format using the infix operator ==> instead of -->.

In the next step, we can initialise the chart top-down (text/1) and make sure that a number of active edges is generated. We do this with the help of a failure-driven loop (foreach/2):

```
chart_parser([],[_,0,_]) :-
  LHS = text([A,B,C,D,E,F,G,H,I]),
  init_chart(0,LHS).

init_chart(V0,LHS) :-
  foreach(rule(LHS,RHS),
    init_edge(V0,V0,LHS,[],RHS)).

init_edge(V0,V0,LHS,Fnd,RHS) :-
  edge(V0,V0,LHS,Fnd,RHS), !.

init_edge(V0,V0,LHS,Fnd,[RHS|RHSs]) :-
  assert_edge(V0,V0,LHS,Fnd,[RHS|RHSs]),
  foreach(rule(RHS,RHS2),
    init_edge(V0,V0,RHS,[],RHS2)),
  update_lah_info(RHS).

foreach(X,Y) :- call(X), once(Y), fail.
foreach(X,Y) :- true.
```

This failure-driven loop calls the transformed DCG rules via the following rules:

```
rule(LHS,RHS)   :- ( LHS ==> RHS ).
rule(LHS,[])    :- ( LHS ==> [] ).
```

and creates a set of active edges for the top-level grammar rules that start and end at vertex 0. Additionally, this initialisation process asserts the initial lookahead information RHS that is available as the first element in the lists of remaining categories [RHS|RHSs]. This is done with the help of the predicate update_lah_info/1 that works in a similar way as the one introduced for the DCG in the last section.

Once the chart is initialised and the initial lookahead information is displayed, the user can select or enter the first word that falls under these classifications. This word is processed by the predicate chart_parser/2, starts at vertex 0 and generates in the case of an atomic word an inactive edge for each instance in the lexicon:

```
chart_parser(Word,[_,V1,V2]) :-
  start_chart(V1,V2,Word).

start_chart(V1,V2,Word) :-
  foreach(word(V1,Word,Cat),
    add_edge(V1,V2,Cat,[],[])).

word(_,Word,Cat) :-
  ( Cat ==> [ lexicon(cat:Cat,wf:Word),
              word(Word) ] ),
  call( lexicon(cat:Cat,wf:Word) ).
```

The chart parser adds these inactive edges to the chart using in a first step the second of the following rules (add_edge/5) and then applies the fundamental rule (fund_rule/4) recursively to inactive and active edges. Note that the first of the following rules checks for duplicates and the third rule applies first the fundamental rule, then predicts new active edges, and finally updates the lookahead information in the subsequent steps:

```
add_edge(V1,V2,LHS,Fnd,RHS) :-
  edge(V1,V2,LHS,Fnd,RHS), !.

add_edge(V1,V2,LHS,Fnd,[]) :-
  assert_edge(V1,V2,LHS,Fnd,[]),
  fund_rule(V1,V2,LHS,[]).

add_edge(V1,V2,LHS,Fnd,[RHS|RHSs]) :-
  assert_edge(V1,V2,LHS,Fnd,[RHS|RHSs]),
  fund_rule(V1,V2,LHS,[RHS|RHSs]),
  predict_active_edges(V2,RHS),
  update_lah_info(RHS).
```

If an inactive edge is asserted by the second rule, then the fundamental rule is applied to every active edge that can make use of this inactive edge:

```
fund_rule(V1,V2,RHS,[]) :-
  foreach(edge(V0,V1,LHS,Fnd,[RHS|RHSs]),
    add_edge(V0,V2,LHS,[RHS|Fnd],RHSs)).
```

If an active edge is asserted in the third rule, then the fundamental rule is first applied to every inactive edge that can make use of this active edge:

```
fund_rule(V1,V2,LHS,[RHS|RHSs]) :-
  foreach(edge(V2,V3,RHS,Fnd,[]),
    add_edge(V1,V3,LHS,[RHS|Fnd],RHSs)).
```

and then the predicate predict_active_edges/2 is applied that looks for each grammar rule (rule/2) that has the category RHS on the left hand side (LHS) and creates new active edges for these rules starting at vertex V2:

```
predict_active_edges(V2,LHS) :-
  foreach(rule(LHS,NewRHS),
    add_edge(V2,V2,LHS,[],NewRHS)).
```

Finally, in a last step, the actual lookahead information (RHS) is updated with the help of the predicate update_lah_info/1.

Note that the rule word/3 can only be used to process atomic words. We use additional rules to deal with compound words. The basic idea is to take a compound word out of the lexicon once the user enters the first token of a compound and write all remaining tokens onto a stack. The first token of these remaining tokens is then displayed as lookahead information and the stack is processed recursively as the user enters more tokens that belong to the compound word. This approach has the advantage that a compound word is looked up only once in the lexicon and the remaining tokens are processed directly from the stack.

7 Evaluation

We have compared the runtimes of the direct DCG approach with the indirect chart parsing approach for generating lookahead information. For the chart parser, we evaluated two settings: (a) the time that it takes to add a new token incrementally if the chart has already been constructed, and (b) the time that it takes to parse a partial sentence from the beginning. In the case of the DCG parser, those two settings coincide since incremental parsing is not possible.

For this comparison, we used the AceWiki grammar together with a test suite of 100 sentences. The longest sentence in this test suite consists of 51 tokens, the shortest sentence of 4 tokens, and the average sentence length is 14 tokens. Many of these sentences contain quite complex syntactic structures, among them embedded relative clauses, negation, coordination, and number restriction, for example:

- Sydney is a part of a country that is not a landlocked country and that borders at least two seas.

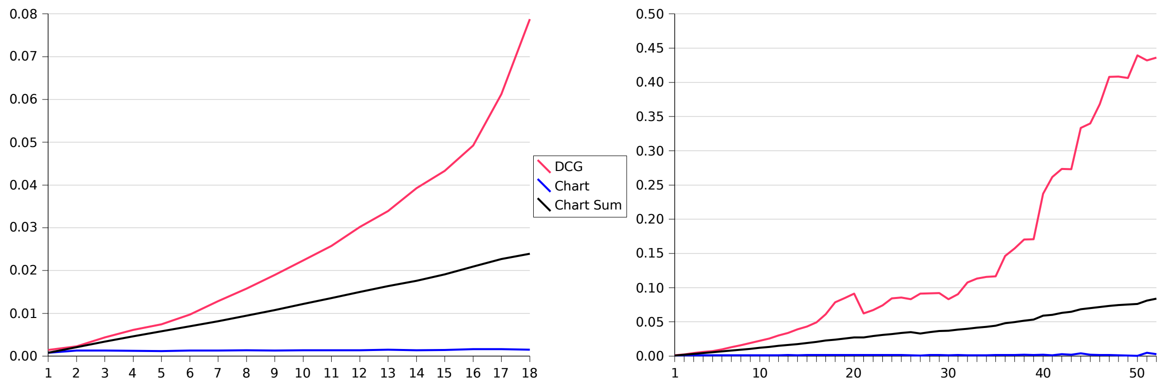


Figure 3: These two graphics show the average runtimes of the three parsing approaches in seconds (y-axis) for the different sentence positions (x-axis) up to position 18 (on the left) and 52 (on the right), respectively.

It turned out that the **average** processing times for generating lookahead information are: 1.3 ms for the chart parsing approach if only the last token has to be processed; 13.1 ms for the chart parsing approach if all of the partial sentence is parsed; and 31.3 ms for the DCG approach. These values where retrieved on a standard 2 GHz *Intel Core Duo* machine using SWI Prolog[6]. Not surprisingly, the latter two values highly depend on the number of tokens in a partial sentence. Figure 3 shows the average time values by sentence position (i.e. number of tokens) for the three approaches. We get very nice curves up to about 18 tokens, and then the DCG curve gets shaky. This is due to the fact that only 25% of the sentences have more than 18 tokens, and thus the sample gets more and more unreliable.

The chart parsing approach requires approximately constant time for each new token, and thus approximately linear time is required if the sentence has to be parsed from the beginning. In both settings, the chart parser performs better than the DCG parser. Surprisingly, not even the processing of the first token in a sentence is faster when we use the DCG instead of the chart parser (1.4 ms versus 0.8 ms). The theoretical worst-case time complexity of the DCG approach is cubic but on average a sentence of 15 tokens can be processed with the AceWiki grammar within 43.3 ms. That means that the DCG approach is still practically useful although theoretically not exhilarating.

[6]`http://www.swi-prolog.org/`

8 Conclusions

In this paper we argued for controlled natural language (CNL) interfaces using predictive text editors to combine human-readability and usability with machine processability. As an example, we presented AceWiki, a semantic wiki, that uses a relatively complex grammar that is based on a subset of Attempto Controlled English (ACE) which is a machine-oriented CNL. A predictive text editor is used to enforce the restrictions of the CNL and to guide the user step-by-step via lookahead information that is generated on the fly while a sentence is written. We have explored two different approaches to generate this lookahead information: the first approach uses the DCG directly and is computationally costly but very easy to implement; the second approach relies on a chart parser and is computationally cheap but requires a bit more work to implement. We showed how the two approaches can be implemented in Prolog, and we compared those implementations. The chart parser performs particularly well if incremental parsing is applied. In any case, the chart parser performs considerably better than the DCG parser. However, the parsing times of the DCG approach are still within a reasonable range to be used in practical applications.

Providing adequate writing support is important for the acceptance of controlled natural languages. In the future, we will refine the presented techniques and study more sophisticated editing operations that can be applied to an existing text and require only minimal reprocessing.

References

I. Androutsopoulos, G. Ritchie, and P. Thanisch. 1995. Natural Language Interfaces to Databases – An Introduction. In: *Journal of Language Engineering*, 1(1), pp. 29–81.

S. Auer, S. Dietzold, and T. Riechert. 2006. OntoWiki – A Tool for Social, Semantic Collaboration. In: *Proceedings of the 5th International Semantic Web Conference*, pp. 736–749, Springer.

V. R. Chintaphally, K. Neumeier, J. McFarlane, J. Cothren, and C. W. Thompson. 2007. Extending a Natural Language Interface with Geospatial Queries. In: *IEEE Internet Computing*, pp. 82–85.

P. Cimiano, P. Haase, J. Heizmann, and M. Mantel. 2008. ORAKEL: A Portable Natural Language Interface to Knowledge Bases. In: *Data & Knowledge Engineering (DKE)* 65(2), pp. 325–354.

P. Clark, P. Harrison, T. Jenkins, T. Thompson, and R. Wojcik. 2005. Acquiring and Using World Knowledge Using a Restricted Subset of English. In: *Proceedings of FLAIRS'05*, pp. 506–511.

P. Clark, P. Harrison, J. Thompson, R. Wojcik, T. Jenkins, and D. Israel. 2007. Reading to Learn: An Investigation into Language Understanding. In: *Proceedings of AAAI 2007 Spring Symposium on Machine Reading*, pp. 29–35.

A. Copestake, K. Sparck Jones. 1990. Natural Language Interfaces to Databases. In: *Knowledge Engineering Review*, 5(4), pp. 225–249.

N. E. Fuchs, U. Schwertel, and R. Schwitter. 1998. Attempto Controlled English – Not Just Another Logic Specification Language. In: *Proceedings of LOPSTR'98*, pp. 1–20.

N. E. Fuchs, K. Kaljurand, T. Kuhn. 2008. Attempto Controlled English for Knowledge Representation. In: *Reasoning Web, Fourth International Summer School 2008*, LNCS 5224, pp. 104–124.

G. Gazdar, C. Mellish. 1998. *Natural Language Processing in Prolog*. An Introduction to Computational Linguistics, Addison-Wesley, 1989.

C. Hallett, R. Power, and D. Scott. 2007. Composing Questions through Conceptual Authoring. In: *Computational Linguistics*, 33(1), pp. 105–133.

F. Hielkema, C. Mellish, P. Edwards. 2008. Evaluating an Ontology-Driven WYSIWYM Interface. In: *Proceedings of INLG 2008*, pp. 138–146.

K. Kaljurand. 2007. *Attempto Controlled English as a Semantic Web Language*. PhD thesis, Faculty of Math. and Computer Science, University of Tartu.

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

M. Krötzsch, D. Vrandecic, M. Völkel, H. Haller, and R. Studer. 2007. Semantic Wikipedia. In: *Journal of Web Semantics*, 5/2007, pp. 251–261, Elsevier.

T. Kuhn. 2008. AceWiki: A Natural and Expressive Semantic Wiki. In: *Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. CEUR Workshop Proceedings.

T. Kuhn. 2008. AceWiki: Collaborative Ontology Management in Controlled Natural Language. In: *Proceedings of the 3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings.

B. Motik, P. F. Patel-Schneider, I. Horrocks. 2008. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft, 11 April 2008. `http://www.w3.org/TR/owl2-syntax/`

F. C. N. Pereira, S. M. Shieber. 1987. *Prolog and Natural-Language Analysis*. CSLI, Lecture Notes, Number 10.

R. Power, D. Scott, and R. Evans. 1998. What You See Is What You Meant: direct knowledge editing with natural language feedback. In: *Proceedings of ECAI 98*, Brighton, UK.

S. Schaffert. 2006. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: *Proceedings of the First International Workshop on Semantic Technologies in Collaborative Applications (STICA06)*.

R. Schwitter. 2002. English as a Formal Specification Language. In: *Proceedings of DEXA 2002*, September 2-6, Aix-en-Provence, France, pp. 228–232.

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

R. Schwitter, M. Tilbrook. 2008. Meaningful Web Annotations for Humans and Machines using Controlled Natural Language. In: *Expert Systems*, 25(3), pp. 253–267.

J. F. Sowa. 2004. Common Logic Controlled English. *Draft*, 24 February 2004. `http://www.jfsowa.com/clce/specs.htm`

H. R. Tennant, K. M. Ross, R. M. Saenz, C. W. Thompson, and J. R. Miller. 1983. Menu-Based Natural Language Understanding. In: *Proceedings of the 21st Meeting of the Association for Computational Linguistics* (ACL), MIT Press, pp. 51–58.

R. H. Tennant, K. M. Ross, and C. W. Thompson. 1983. Usable Natural Language Interfaces Through Menu-Based Natural Language Understanding. In: *CHI'83: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 154–160, New York, NY.

C. Thompson, P. Pazandak, and H. Tennant. 2005. Talk to Your Semantic Web. In: *IEEE Internet Computing*, 9(6), pp. 75–78.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 55-63

Classification of Verb-Particle Constructions with the Google Web1T Corpus

Jonathan K. Kummerfeld and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
`{jkum0593,james}@it.usyd.edu.au`

Abstract

Manually maintaining comprehensive databases of multi-word expressions, for example Verb-Particle Constructions (VPCs), is infeasible. We describe a new type level classifier for potential VPCs, which uses information in the Google Web1T corpus to perform a simple linguistic constituency test. Specifically, we consider the fronting test, comparing the frequencies of the two possible orderings of the given verb and particle. Using only a small set of queries for each verb-particle pair, the system was able to achieve an F-score of 75.7% in our evaluation while processing thousands of queries a second.

1 Introduction

Creating comprehensive linguistic resources manually is an expensive and slow process, making it infeasible for maintaining up-to-date resources of constantly evolving linguistic features, such as Multi-Word Expressions (MWEs). These resources are crucial for a range of Natural Language Processing (NLP) tasks, such as accurate parsing. Identification of MWEs does not prevent the production of syntactically accurate parses, but the extra information can improve results. Since manually creating these resources is a challenge, systems are needed that can automatically classify expressions accurately. Also, because these systems will be running during parsing, they need to be fast to prevent the creation of an additional bottleneck.

Here we focus on Verb-Particle Constructions (VPCs), a type of MWE composed of a verb and a particle. Villavicencio (2003a) showed that VPCs are poorly covered by corpus data and constantly growing in number, making them a suitable candidate for an automatic classification system. Previous work on VPCs has mainly focused either on their compositionality (McCarthy et al., 2003), or on using sophisticated parsers to perform extraction from corpora (Baldwin and Villavicencio, 2002). While parser based methods have been very successful (Kim and Baldwin, 2006) they rely on contextual knowledge and complex processing. The Web has been used as a corpus previously by Villavicencio (2003b), who used search engines as a source of statistics for classification, but her aim was to create new resources, rather than a tool that can be used for identification. We have constructed a high-throughput query system for the Google Web1T data, which we use to collect information to perform the fronting linguistic constituency test. The results of this process are used to train a classifier, which can then quickly and accurately classify a given verb-particle pair.

The Google Web1T corpus is over twenty-four gigabytes of plain text in compressed form, making it infeasible to store the entire data-set in memory and perform queries directly. To handle the data we have aggregated the frequency counts for n-grams by using templates that contain a mixture of wild-cards and words, where the words are all possibilities taken from three sets. The new data this process produces is stored in a hash-table distributed across several computers, making fast queries possible. These queries provide the frequency of templates such as `verb ? particle` and `particle ? verb`, which can then be compared as a form of the fronting test. By comparing the frequency of these templates we are able to construct classifiers for verb-particle pairs with an accuracy of 74.7%, recall of 74.7%, and a precision of 76.8%. These results indicate that this method is a promising source of information for fast on-line classification of verb-particle pairs as VPCs.

2 Background

VPCs are MWEs composed of a verb and a particle, where particles are derived from two categories, prepositions and spatial adverbs (Quirk et al., 1985). However, most research on VPCs has focused on prepositions only, because they are more productive. Semantically, the compositionality of multi-word expressions varies greatly. Often they have the property of paraphrasability, being replacable by a single verb of equivalent meaning, and exhibit prosody, having a distinctive stress pattern.

Previous work concerning VPCs can be broadly divided into two groups, compositionality analysis, and classification. Our work is entirely concerned with classification, but we will briefly describe previous work on compositionality as the two areas are closely linked.

2.1 Compositionality

Determining how much the simplex meaning of individual words in a MWE contribute to the overall meaning is challenging, and important for producing semantically correct analysis of text. A range of methods have been considered to differentiate examples based on their degree of semantic idiosyncrasy.

Initial work by Lin (1999) considered the compositionality of MWEs by comparing the distributional characteristics for a given MWE and potentially similar expressions formed by synonym substitution. This was followed by Bannard et al. (2003), who used human non-experts to construct a gold standard dataset of VPCs and their compositionality, which was used to construct a classifier that could judge whether the two words used contributed their simplex meaning. McCarthy et al. (2003) used an automatically acquired thesaurus in the calculation of statistics regarding the compositionality of VPCs and compared their results with statistics commonly used for extracting multiwords, such as latent semantic analysis. Bannard (2005) compared the lexical contexts of VPCs and their component words across a corpus to determine which words are contributing an independent meaning.

Light Verb Constructions (LVCs) are another example of an MWE that has been considered in compositionality studies. The challenge of distinguishing LVCs from idioms was considered by Fazly et al. (2005), who proposed a set of statistical measures to quantify properties that relate to the compositionality of an MWE, ideas that were then extended in Fazly and Stevenson (2007).

Recently, Cook and Stevenson (2006) addressed the question of which sense of the component words is being used in a particular VPC. They focused on the contribution by particles and constructed a feature set based on the properties of VPCs and compared their effectiveness with standard co-occurrence measurements.

2.2 Classification

A range of methods for automatic classification of MWEs have been studied previously, in particular the use of parsers, applying heuristics to n-grams, and search engine queries.

Possibly the earliest attempt at classification, was by Smadja (1993), who considered verb-particle pairs, separated by up to four other words, but did not perform a rigorous evaluation, which prevents us from performing a comparison.

Recent work has focused on using parsers over raw text to gain extra information about the context of the verb-particle pair being considered. In particular, the information needed to identify the head noun phrase (NP) for each potential VPC. Early work by Blaheta and Johnson (2001) used a parsed corpus and log-linear models to identify VPCs. This was followed by Baldwin and Villavicencio (2002) who used a range of parser outputs and other features to produce a better informed classifier. Other forms of linguistic and statistical unsupervised methods were considered by Baldwin (2005), such as Pointwise Mutual Information. This work was further extended by Kim and Baldwin (2006) to utilise the sentential context of verb-particle pairs and their associated NP to improve results.

The closest work to our own in terms of the corpus used is that of Villavicencio (2003b), in which the web is used to test the validity of candidate VPCs. Also, like our work, Villavicencio does not consider the context of the verb and particle, unlike the parser based methods described above. A collection of verb-particle pairs was generated by combining verbs from Levin's classes (Levin,

1993) with the particle up. Each potential VPC was passed to the search engine Google to obtain an approximate measure of their frequency, first on their own, then in the form `Verb up for` to prevent the inclusion of prepositional verb forms. The measurement is only approximate because the value is a document count, not an actual frequency as in the Web1T data. This led to the identification of many new VPCs, but the results were not evaluated beyond measurements of the number of identified VPCs attested in current resources, making comparison with our own work difficult.

When a verb-particle pair is being classified the possibilities other than a VPC are a prepositional verb, or a free verb-preposition combination. A variety of tests exist for determining which of these a candidate verb-particle pair is. For the transitive case, Baldwin and Villavicencio (2002) applied three tests. First, that VPCs are able to undergo particle alternation, providing the example that hand in the paper and hand the paper in are both valid, while refer to the book is, but *refer the book to is not. Second, that pronominal objects must be expressed in the split configuration, providing the example that hand it in is valid, where as *hand in it is not. And finally, that manner adverbs cannot occur between the verb and particle. The first two of these tests are context based, and therefore not directly usable in our work, and the third is particularly specific. Instead, for now we have focused on the 'fronting' constituency test, which involves a simple rearrangement of the phrase, for example hand in the paper would be compared to *in the paper hand.

2.3 The Google Web1T corpus

The Google Web1T corpus is an extremely large collection of n-grams, extracted from slightly more than one trillion words of English from public web pages (Brants and Franz, 2006). This dataset poses new challenges for data processing systems, as it is more than twenty-four gigabytes in compressed form.

The corpus contains n-grams up to 5-grams, such as in Table 1. This example considers the pair of words 'ferret' and 'out', showing all entries in the corpus that start and end with the two words, excluding those that contain non-alphabetical characters.

N-gram	Frequency
ferret out	79728
out ferret	74
ferret her out	52
ferret him out	342
ferret information out	43
ferret is out	54
ferret it out	1562
ferret me out	58
ferret that out	180
ferret them out	1582
ferret these out	232
ferret things out	58
ferret this out	148
ferret you out	100
out a ferret	63
out of ferret	71
out the ferret	120
ferret it all out	47
ferret these people out	60
ferret these projects out	52
out of the ferret	54
ferret lovers can ferret out	45
out a needing shelter ferret	63

Table 1: N-grams in the Web1T corpus for the VPC ferret out.

One approach to this dataset is to simply treat it as a normal collection of n-gram frequencies, scanning the relevant sections for answers, such as in Bergsma et al. (2008) and Yuret (2007). Another approach is used by Talbot and Brants (2008), pruning the corpus to a third of its size and quantising the rest, reducing the space used by frequency counts to eight bits each. These methods have the disadvantages of slow execution and only providing approximate frequencies respectively.

Hawker et al. (2007) considered two methods for making practical queries possible, preprocessing of the data, and pre-processing of queries. The first approach is to reduce the size of the dataset by decreasing the resolution of measurements, and by accessing n-grams by implicit information based on their location in the compressed data, rather than their actual representation. While this avoids the cost of processing the entire dataset for each query, it does produce less accurate results. The second method described is intelligent batching queries, then performing a single pass through the data to answer them all.

The Web1T corpus is not the only resource for n-gram counts on the web. Lapata and Keller (2005) use Web counts, the number of hits for a search term, as a measure of the frequency of n-grams. This approach has the disadvantage that the frequency returned is actually a document frequency, where as the Web1T values are counts of the occurrences of the actual n-gram, including repetitions in a single page.

2.4 Weka

Weka is a collection of machine learning algorithms for data mining tasks (Witten and Frank, 2005). From the range of tools available we have used a selection of the classifiers: ZeroR, OneR, Id3, J48, Naive Bayes and Multilayer Perceptron.

As a baseline measure we used the ZeroR classifier, which always returns the mode of the set of nominal options. Next we considered the OneR classifier, which compares the information gain of each of the features and chooses the most effective. Two decision tree classifiers were considered, Id3 (Quinlan, 1993b) and J48 (Quinlan, 1993a). Both initially construct the tree in the same way, using information gain to choose the feature to split on next, until either no features remain, or all samples are of the same class. The difference is that J48 attempts to prune the tree.

We also considered two non-decision based classifiers, Naive Bayes and Multilayer Perceptron. The Naive Bayes classifier assumes features are independent and applies Bayes' Theorem to calculate the probability of an example belonging to a particular class. The Multilayer Perceptron uses multiple layers of sigmoid nodes, whose links are adjusted during training by back-propagation of corrections (Rumelhart et al., 1986).

3 Classifying verb-particle pairs

Our aim was to construct a system capable of classifying a given verb-particle pair as a potential VPC or not. To do this we currently apply the fronting test to the verb-particle pair. We describe the classification given by the system as 'potential' because the system only uses the verb-particle pair and not the specific context being considered. The system also needed a simple interface for making fast queries.

The fronting test is a linguistic constituency test that gauges whether words function as a single unit in a phrase by breaking the phrase in half between the words and swapping the order of the two halves (Quirk et al., 1985). In the case of VPCs this means the following rearrangement (note that NP_2 may not always be present):

- NP_1 Verb Particle NP_2
- Particle NP_2 NP_1 Verb

If the rearranged form of a phrase is also valid it implies the verb and particle are not functioning as a single unit in the phrase, and so we do not classify them as a VPC.

In this structure there are two forms of verbs to consider. For the intransitive case a VPC always results (Baldwin and Villavicencio, 2002), for example:

- He gets around.
- *Around he gets.

However, in the transitive case there are three possibilities, for example:

- The mafia ran down Joe.
- *Down Joe the mafia ran.

- What did you refer to?
- To what did you refer?

- I walked to the house.
- To the house, I walked.

The last two examples are not VPCs, but rather examples of a prepositional verb and a free verb-preposition combination respectively. Note that both orders are valid for these, where as for the VPCs the second is not.

We use the frequency counts in the Web1T data to measure the validity of particular orderings. For this purpose the meaning of the absolute value returned is difficult to quantify, so instead we compare the frequencies of multiple arrangements of the same verb-particle pair. The more common arrangement is treated as being more valid. The exact form of these comparisons is described in the following sections.

4 Implementation

4.1 Structure

At the core of our system is a hash table that allows random access to the parts of the Web1T data that are relevant to our queries. The table uses linear probing based on a 64bit key that is a numerical representation of each word pattern, interpreted in base thirty-seven. The initial position considered is the key value modulo the size of our table (which may vary to allow support for different amounts of system RAM). While there is the possibility of collisions, the probability of their occurrence is extremely small, and none were observed during testing.

Word patterns are used to generate the key, and at the location specified by the key (or a position that is probed to) we store the complete key and four integers. The system is designed to process any string to generate the key, rather than specifically VPCs, and the four integers stored are returned directly, to be manipulated as desired. We use these four integers to hold aggregated counts and the string used is based on word patterns containing wildcards.

The word patterns used to create the keys may contain two types of symbols, a wildcard, or a set reference. Wildcards represent any word not in the given sets, where we do not include punctuation, numbers or sentence boundaries as words. Set references indicate that the word at that position must come from one of the provided sets. Currently the system handles up to three sets, each of which can be referenced at most once. Also note that patterns with leading or trailing wildcards do not need to be considered, since they are already included in the count for the equivalent n-gram without the leading or trailing wildcards.

For two sets, the four patterns used are as follows, where _ represents a wildcard, and `A` and `B` are set references:

1. `A B`
2. `A _ B`
3. `A _ _ B`
4. `A _ _ _ B`

For three sets we cannot store the frequency of each pattern separately as there are six possible patterns, but only four spaces in each entry of the table. We chose to aggregate the values further, accepting the following patterns:

1. `A B C`
2. `A B _ C or A B _ _ C`
3. `A _ B C or A _ _ B C`
4. `A _ B _ C`

All data in the Web1T corpus that does not conform to one of these patterns is ignored. This means the size of the hash table needed, and therefore the amount of memory needed, depends entirely on the sizes of the word sets provided.

The hash table, word sets, and their interface, are written in C++, but are wrapped up inside a Python extension. This python extension is used by a server-client interface, where each server holds a single hash table that has all the aggregated counts for words within a specific alphabetic range. To perform queries a client connects to a set of servers, receives their ranges, and then requests frequency counts directly from the appropriate server. This structure effectively increases the size of the system memory that is accessible, but also has the side effect that multiple clients can query the system at the same time at high speed.

4.2 Statistics

We performed our experiments using three sets of words, verbs, particles and pronouns. The sizes and sources of the sets were:

- Verbs - 19,046 - all words with a VB based tag in the Wall Street Journal section of the Penn Treebank
- Possible Particles - 257 - all words with either an IN or RP tag in the Wall Street Journal section of the Penn Treebank
- Pronouns - 77 - from Wikipedia

This leads to 2,261,407,764 possible permutations of the words in the word sets. The storage cost for each permutation is 24 bytes, eight of which is for the 64bit key, and the other sixteen bytes are for the four integers, which occupy four bytes each. However, because most of these permutations are not found in the Web1T data, a total of only approximately 2.8 Gigabytes of system memory was required.

Pattern	Frequency
`Verb Particle`	2 436 566
`Verb _ Particle`	747 492
`Verb _ _ Particle`	78 569
`Verb _ _ _ Particle`	19 549
`Particle Verb`	2 606 582
`Particle _ Verb`	2 326 720
`Particle _ _ Verb`	68 540
`Particle _ _ _ Verb`	14 016

Table 2: Word patterns and example aggregated frequencies for the VPC hand in.

Since one of our aims was to produce a quick method for classification, the speed of the system is also worth considering. Once running, the servers were able to handle between five and ten thousand queries per second. The time to start the servers, which includes reading all of the Web1T data, was between sixty and ninety minutes. However, this time could be drastically reduced for subsequent runs by writing the constructed hash table out to disk and reading it directly into memory the next time the system needs to be started.

4.3 Hash table queries for VPCs

Once the hash table is constructed the system is able to take two or three words and return the four values corresponding to their position in the hash table (where the words are considered in the order they are given). A simplified form of the fronting test considers the patterns in Table 2.

The word patterns in the top half of Table 2 correspond to $\texttt{NP}_1$ `Verb Particle` $\texttt{NP}_2$. We have considered the cases with wildcards between the verb and particle since most VPCs can occur in the split configuration. The patterns in the bottom half of Table 2 correspond to `Particle` $\texttt{NP}_2$ $\texttt{NP}_1$ `Verb`. We allow the case with a single wildcard to cover examples without a second `NP`. To get the vales for these patterns the system is queried twice, once with `Verb Particle` and then with `Particle Verb`.

As mentioned previously, it is difficult to quantify the absolute values returned by these queries, but by comparing them we are able to measure the relative validity of the two orderings, as described in the following section.

Name	Comparison
vp-1	`v ? ? ? p > p _ _ ? v`
vp-2	`v p > p _ v`
nv_p-1	`n v ? ? p > p n v`
nv_p-2	`n v ? ? p > p ? ? n v`
n_vp-1	`n _ ? v ? p > p n _ ? v`
n_vp-2	`n _ ? v ? p > p ? n _ ? v`

Table 3: Comparisons for judging whether a verb and particle form a VPC.

One problem with this comparison is that the contributions to the values could be coming from different uses of the pair, such as in Table 1, where ferret is out and ferret it out will both contribute to the frequency for the pattern `Verb _ Particle`. To deal with this we added queries that used the three set patterns. As the third set we used pronouns, which can approximate single word NPs. This method does have the drawback that frequencies will be lower, leading to more unattested examples, but does provide extra information in most cases.

5 Results

For the purposes of classification we created a set of questions that have true or false answers and can be answered based on queries to our system. In many of these cases we consider the sum of a collection of values from our hash table. To keep the descriptions here concise we have used the symbol '?' to denote the optional presence of another wildcard. The value for patterns containing '?'s is the sum of all the possible patterns, with or without wildcards at those positions. Also, rather than using the entire words, we have used 'v' for Verbs, 'p' for Particles and 'n' for pronouns.

To produce a set of invalid VPCs we used our lists of verbs and particles to generate a random list of two thousand verb-particle pairs not in our set of VPCs. The set of two thousand true VPCs was constructed randomly from a large set containing 116 from McCarthy et al. (2003) and 7140 from Baldwin and Villavicencio (2002).

For every pair a feature vector containing the results of these comparisons was created, as demonstrated in Table 4, containing the answers to the questions described previously. Weka was

Test	Comparison	Result
vp-1	3 282 176 > 82 556	True
vp-2	2 436 566 > 2 326 720	True
nv_p-1	435 822 > 1 819 931	False
nv_p-2	435 822 > 1 819 931	False
n_vp-1	0 > 0	False
n_vp-2	0 > 0	False

Table 4: Example comparisons for the VPC hand in.

then used to generate and evaluate a range of classification techniques. All methods were applied with the standard settings and using ten fold cross-validation.

6 Discussion

This lightweight method of classification does not match up to results from parser based methods, such as Kim and Baldwin (2006), who acheived an F-score of 97.4%. However, this high performance relies on context awareness and only applied to the identification of frequent VPC examples. In contrast, by drawing on the Web1T dataset our system overcomes the data sparseness problem. The cost of attempting to classify without knowledge of context is that we apply a linguistic poor approach, achieving lower performance, but there is plenty of scope for extension. The system as it stands is very flexible and can easily scale to support larger sets of words, making a broader range of queries possible. Also, the post-processing of query results could become more sophisticated, and the classifiers used could be tweaked to be more effective. The system itself could also be modified to store other information, making different types of queries possible.

One challenge that remains unaddressed is how to handle multiple uses of the same word pair. For example, the following are both valid:

- The deadline is coming up.
- Joe is an up and coming athlete.

In these two cases the words coming and up have different semantic meanings, but both forms will contribute to the frequency counts for the word pair. While this does skew the results of our queries, we should be able to limit the effect by using the third set to constrain how the words are being used.

Despite the support for three sets of words, we only really have flexibility in one of them, as we must use the other two for Verbs and Particles. Here we chose Pronouns because they could act as single word NPs, allowing us to perform the fronting test more effectively. However, there are other linguistic tests for VPCs. In particular, two of the tests performed by Baldwin and Villavicencio (2002) could be adapted to our system. One is that transitive VPCs must be expressed in the split configuration when the second NP is pronominal, such as hand it in, which is valid, and *hand in it, which is not. Our third set already contains pronouns, all that needs to be done is a comparison of the frequency of `Verb Pronoun Particle` and `Verb Particle Pronoun`, giving a measure of which is more valid. The other test is that manner adverbs cannot occur between the verb and the particle. This could be tested by expanding our third set to include manner adverbs, and then comparing the frequency of `Verb Manner Adverb Particle` and `Verb Particle`.

So far we have only considered a small number of ways to combine the results of the queries to produce features for classifiers. All of the features we considered here were true/false answers to simple questions, constructed intuitively, but without much experimentation. By breaking up our training set and using a part of it to explore the statistical properties of the results from queries we may be able to identify unexpected, but effective, discriminating factors. We may also be able to improve performance by considering a broader range of classifiers, and by adjusting their input parameters to suit our task.

Finally, the system itself could be modified to store more information and to support other types of queries. The corpus does contain sentence boundary markers, and we could use capitalisation to extend this information approximately without wasting a word in the n-gram. Currently these are entirely ignored, except when a sentence boundary occurs in the middle of an n-gram, in which case the n-gram is excluded from the frequency count. Also, because the current system uses linear probing, the entire hash table is stored

Classifier	Accuracy	Precision	Recall	F score
ZeroR	50.0%	25.0%	50.0%	33.3%
OneR	74.4%	76.4%	74.4%	75.4%
Id3	74.7%	76.8%	74.7%	75.7%
J48	74.4%	76.4%	74.4%	75.4%
Naive Bayes	74.7%	76.8%	74.7%	75.7%
MultilayerPerceptron	73.4%	73.6%	73.4%	73.5%

Table 5: Results using Weka on simple query answers.

as a single block of memory. This means the fread and fwrite functions could easily be used to save it on disk, which would drastically reduce the system restart time (assuming the same word sets are being used).

7 Conclusion

We have constructed a system for quickly querying the Web1T corpus, providing the information needed to perform the fronting linguistic constituency test. By performing this test on potential VPCs, on their own, and in combination with a range of pronouns, we produce a feature vector that can be used to construct a classifier. The classifiers produced are accurate and fast, as the only information they need can be quickly accessed from the Web1T corpus by our system. The variation in performance between classifiers is small, but the best performance is by the Id3 and Naive Bayes classifiers, which have a recall of 74.7%, and precision of 76.8%.

A range of other linguistic tests exist, which our system could be extended to support. Adding this extra information to our feature vectors and exploring classification methods further could lead to improved performance. Another area for further investigation is the interpretation of the raw frequency counts and the way comparisons are performed.

Our system is capable of scaling to support a larger proportion of the Web1T data, while retaining the ability to respond quickly to many queries. Also, by adjusting the word sets provided and/or the templates that are used to aggregate frequency counts, our system could be altered to support different types of queries. This flexibility means it could be useful in other tasks that involve querying the Web1T corpus.

8 Acknowledgement

We would like to thank the anonymous reviewers for their helpful feedback and corrections. J. K. K. thanks The University of Sydney for the support of a Merit Scholarship and the Talented Student Program that enabled this work.

References

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: a case study on verb-particles. In *Proceedings of the 2002 Conference on Natural Language Learning*, pages 1–7, Taipei, Taiwan, August.

Timothy Baldwin. 2005. Looking for prepositional verbs in corpus data. In *Proceedings of the 2005 Meeting of the Asociation for Computational Linguistics: Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 115–126, Colchester, UK, April.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proceedings of the 2003 Meeting of the Asociation for Computational Linguistics: Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 65–72, Sapporo, Japan, July.

Colin Bannard. 2005. Learning about the meaning of verb-particle constructions from corpora. *Journal of Computer Speech and Language*, 19(4):467–478.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Distributional identification of non-referential pronouns. In *Proceedings of the 2008 Meeting of the Asociation for Computational Linguistics: Human Languages Technologies*, pages 10–18, Columbus, Ohio, June.

Don Blaheta and Mark Johnson. 2001. Unsupervised learning of multi-word verbs. In *Proceedings of the 2001 Meeting of the Asociation for Computational Linguistics: Workshop on Collocation the Compu-*

tational Extraction, Analysis and Exploitation of Collocations, pages 54–60, Toulouse, France, July.

Thorsten Brants and Alex Franz. 2006. Web1t 5-gram corpus version 1.1. Technical report, Google Research.

Paul Cook and Suzanne Stevenson. 2006. Classifying particle semantics in English verb-particle constructions. In *Proceedings of the 2006 Meeting of the Association for Computational Linguistics and the International Committee on Computational Linguistics Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 45–53, Sydney, Australia, July.

Afsaneh Fazly and Suzanne Stevenson. 2007. Distinguishing subtypes of multiword expressions using linguistically-motivated statistical neasures. In *Proceedings of the 2007 Meeting of the Asociation for Computational Linguistics: Workshop on A Broader Perspective on Multiword Expressions*, pages 9–16, Prague, Czech Republic, June.

Afsaneh Fazly, Ryan North, and Suzanne Stevenson. 2005. Automatically distinguishing literal and figurative usages of highly polysemous verbs. In *Proceedings of the 2005 Meeting of the Association for Computational Linguistics: Workshop on Deep Lexical Acquisition*, Ann Arbor, USA, July.

Tobias Hawker, Mary Gardiner, and Andrew Bennetts. 2007. Practical queries of a massive n-gram database. In *Proceedings of the 2007 Australasian Language Technology Workshop 2007*, pages 40–48, Melbourne, Australia, December.

Su Nam Kim and Timothy Baldwin. 2006. Automatic identification of english verb particle constructions using linguistic features. In *Proceedings of the 2006 Meeting of the Asociation for Computational Linguistics: Workshop on Prepositions*, pages 65–72, Trento, Italy, April.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–33.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 1999 Annual Meeting of the Association for Computational Linguistics*, pages 317–324, College Park, Maryland, USA.

Diana McCarthy, Bill Keller, , and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the 2003 Meeting of the Asociation for Computational Linguistics: Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, Sapporo, Japan, July.

John Ross Quinlan. 1993a. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

John Ross Quinlan, 1993b. *Induction of decision trees*, pages 349–361. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, London.

David Rumelhart, Geoffrey Hinton, and Ronald Williams, 1986. *Learning internal representations by error propagation*, pages 318–362. MIT Press, Cambridge, MA, USA.

Frank Smadja. 1993. Retrieving collocations from text: Xtract. *Journal of Computational Linguistics*, 19(1):143–177.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of the 2008 Meeting of the Asociation for Computational Linguistics: Human Languages Technologies*, pages 505–513, Columbus, Ohio, June.

Aline Villavicencio. 2003a. Verb-particle constructions and lexical resources. In *Proceedings of the Meeting of the Asociation for Computational Linguistics: 2003 workshop on Multiword expressions*, pages 57–64, Sapporo, Japan, July.

Aline Villavicencio. 2003b. Verb-particle constructions in the world wide web. In *Proceedings of the 2003 Meeting of the Association for Computational Linguistics: Workshop on the Linguistic Dimensions of Prepositions and their use in Computational Linguistics Formalisms and Applications*, Sapporo, Japan, July.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Fransisco, 2nd edition.

Deniz Yuret. 2007. KU: Word sense disambiguation by substitution. In *Proceedings of the 2007 International Workshop on Semantic Evaluations*, Prague, Czech Republic, June.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 64-72

Requests and Commitments in Email are More Complex Than You Think: Eight Reasons to be Cautious

Andrew Lampert
CSIRO ICT Centre
Locked Bag 17
North Ryde 1670
Australia
Andrew.Lampert@csiro.au

Robert Dale
Centre for Language Technology
Macquarie University 2109
Australia
rdale@ics.mq.edu.au

Cécile Paris
CSIRO ICT Centre
Locked Bag 17
North Ryde 1670
Australia
Cecile.Paris@csiro.au

Abstract

Many workplace tasks are managed through email communication, involving the exchange of requests and commitments. Our aim is to build a tool that can automatically identify and manage such requests and commitments. A detailed analysis of real data, however, reveals a range of interesting edge cases that make even human annotation of training data difficult. In this paper, as an important step in the development of annotation guidelines for wider use in the growing email processing community, we identify eight categories of problematic data and propose how they should be handled in annotation and extraction tasks.

1 Introduction

Our aim is to create tools that assist email users by automatically detecting requests and commitments in incoming and outgoing email. The motivation for this is well explained by observations from ethnographic research into the use of electronic messaging in the workplace (Murray, 1991):

> [Managers] would like to be able to track outstanding promises they have made, promises made to them, requests they've made that have not been met and requests made of them that they have not fulfilled.

Other studies have also highlighted that people routinely use email for managing requests and commitments (e.g., (Mackay, 1988; Ducheneaut and Bellotti, 2001)), but struggle to give appropriate attention to requests and commitments that require action or response because they are buried in their email (Whittaker and Sidner, 1996). More recent studies of task-focused email usage have also identified problems with "keeping track of lots of concurrent actions: One's own to-dos and to-dos one expects from others" using existing email clients (Bellotti et al., 2003).

To provide support here, we are working to augment existing email clients with features such as action-oriented summaries of email messages and threads; task-based navigation and visualisations; and dashboards that provide overviews of the state of an email inbox or collection with much greater fidelity than is possible with current tools.

In working towards this goal, we have conducted a series of manual annotation experiments, exploring the level of human agreement that is achievable in identifying requests and commitments in email messages. Previous work has often relied on canonical examples as the basis for simple definitions of requests and commitments. Our experiments have found that this level of detail is insufficient to obtain reliable interannotator agreement. Indeed, we have discovered a range of edge cases not well represented by simple examples or definitions. The influence of factors such as politeness lead to a variety of more complex and indirect requests and commitments than other researchers have assumed. Such complexity is found when classifying requests and commitments at both the message-level and the utterance-level.

Our analysis suggests that the range of these edge cases is not an unstructured collection of widely-varying 'hard cases'; rather, we believe that there are a number of major distinct categories, and that it is useful to identify these categories.

In this paper, we provide just such a categorisation of these edge cases. For each category we propose how the data should be annotated with regard to the presence or absence of requests and commitments. Note that there are two distinct points here: depending on the intended use of the data, one might make different decisions as to whether a request or commitment is present in a given instance, but the categories remain. Our principle focus here is to identify and define the categories, so that they will be appropriately considered and acknowledged in subsequent work; they cannot be 'brushed under the carpet'.

In Section 2 we provide an overview of our definitions of requests and commitments. Section 3 provides a brief overview of the email data we are working with and the annotation experiments we have carried out so far. Section 4 identifies and characterises the eight phenomena that we believe need to be explicitly addressed and indicates how we handle these in our annotation scheme. Section 5 makes some concluding remarks and discusses implications of our analysis for automating the identification of requests and commitments in email.

2 Definitions

In defining requests and commitments, we have looked extensively into previous work that has attempted to classify similar phenomena in email and other media. In (Lampert et al., 2008), we described why none of the existing definitions were suitable for our needs. Briefly, many existing definitions deal only with requests and ignore conditionality (which is very common) as a feature (for example, (Camino et al., 1998; Khosravi and Wilks, 1999; Leuski, 2004)). Others define requests and commitments in terms of specific conversation states, requiring the creation of multiple categories for the same speech act in different stages of a conversation. Often not all of the many combinations of speech acts and conversation states are modeled, resulting in uncodable utterances (Cohen et al., 2004; Goldstein and Sabin, 2006).

Previous work on utterance-level classification (Corston-Oliver et al., 2004) relied on short, simple definitions and canonical examples. These lack the detail and clarity required for unambiguous classification of the complex requests and commitments we find in real-world email. Since our review of related work, Scerri et al. (2008) have noted some similar concerns. Unfortunately, their interannotator agreement for requests and commitments remains low; we believe this could be improved through the careful consideration of the edge cases we outline in this paper.

Conditionality is an important part of our definitions. Conditional requests and commitments require action only if a stated condition is satisfied. Our early annotation experiments, summarised in Section 3 and detailed in (Lampert et al., 2007), show that annotators require guidance about how to classify conditional requests and commitments to achieve even moderate agreement. Others have since replicated this finding (Scerri et al., 2008).

Another issue is the complexity of the relationship between a request or commitment and its possible realisations in surface text. To deal with this, we explicitly distinguish the realisation of a request or commitment from the underlying request or commitment itself. This allows us to adopt the traditional linguistic distinction between direct and indirect speech acts as alternative means of realising requests and commitments. It also allows us to talk of different realisations of the *same* request or commitment, which, as we note in Section 4, is important for dealing with cases where a given request or commitment is stated more than once in a single message.

Below, we give a high-level overview of our definitions for requests and commitments. More detail about the definitions and our treatment of conditionality and the realisation of requests and commitments can be found in (Lampert et al., 2008).

2.1 Requests

We consider a request to be an utterance from an email sender that places an obligation on an email recipient to:

1. Schedule an action, often by adding an entry to a calendar or task list;
2. Perform an action; or
3. Respond with some speech act.

Requests for action, information, permission, confirmation, agreement, evaluation, interpretation, and sympathy (Labov and Fanshel, 1977) can all function as requests. Some linguists have distinguished between speech acts that require a physical response from those that require a verbal or information response (see, for example, (Sinclair and Coulthard, 1975)). We follow Searle's original approach (Searle, 1969) and do not distinguish between physical and verbal responses. We thus explicitly include questions requiring an informational response as requests, since they represent an attempt by the sender to elicit an action from the recipient, in the form of a speech act.

2.2 Commitments

We consider a commitment to be an offer or promise made by an email sender for future action or response from some specified agent. The agent who is committed is often, but not always, the sender. In contrast to previous work, we include as commitments utterances that place an obligation on another person, group of people, or organisation. Such third-party commitments are common in the email corpus we are working with, and we believe at least some of these to be important commitments to capture. A useful, though not conclusive, diagnostic test for identifying a commitment is whether you might expect to find the future action on the responsible agent's task list or calendar.

2.3 Considering the End Goal

As a final consideration in our definitions, we explicitly take account of the *purpose* of identifying and extracting requests and commitments, in the sense of what the end application or intended use of the identified material is.

Intended use is a particularly important consideration, since different types of requests and commitments are likely to be considered relevant for different end uses. Within an email client, for example, the requests and commitments that users would like to have highlighted whilst reading an email message (where a user has access to the entire message content and context) are likely to differ from those that should be extracted and aggregated into a separate task list or task dashboard (where the requests and commitments are removed from their original message content and context). The set of tasks to include in an action-based summary of a message or thread, displayed in the inbox alongside existing message metadata (such as sender, date, subject and so on) would be different again.

			3-way κ	
Experiment	Msgs	Sentences	Request	Commit
Sentences 1	54	310	0.78	0.54
Sentences 2	350	750	0.80	0.74
Messages	100	–	0.84	0.78

Table 1: Agreement from manual annotation experiments

Our goal is *to identify all requests and commitments that would be useful to highlight for a recipient while they are reading an email message*. Using this application scenario as a diagnostic supports our goal of high recall, since this set of tasks is likely to be a superset of the tasks required for other uses.

3 The Data and Annotation Experiments

Our insights in this paper are based on observations from a series of annotation experiments we conducted using data from the Enron email corpus. We employed the database dump of the corpus released by Andrew Fiore and Jeff Heer.[1] This version of the corpus has been processed to remove duplicate email messages and to normalise sender and recipient names, resulting in just over 250,000 email messages without attachments. All data annotated in our experiments is extracted from message bodies.

Table 1 shows an overview of interannotator agreements from our experiments (as Cohen's κ scores). These agreements refer to binary agreement about whether a specific sentence or message contains a request or commitment. The first two experiments annotated sentences, while the third experiment involved annotations at the message level.

The *Sentences 1* experiment presented in (Lampert et al., 2007) used guidelines similar to many of those found in previous work. The resulting agreement was moderate for requests, and poor for commitments. *Sentences 2* gave more explicit guidance for annotating conditional and indirect requests and commitments, which led to increased agreement, in

[1]Available at http://bailando.sims.berkeley.edu/enron/enron.sql.gz

particular for commitments. The *Messages* experiment was conducted to explore the effect of the different unit size (message *vs.* sentence) on agreement. Agreement for both requests and commitments increased over the sentence-level experiments. We discuss this experiment further in Section 4.1. All experiments were performed over randomly-sampled messages drawn from the Enron corpus with three annotators.

4 Edge Case Phenomena in Email

We report here on systematic challenges discovered through manually annotating requests and commitments in email messages. We organise our discussion around classes of phenomena. We argue that these must be carefully considered when attempting to classify requests and commitments in email, whether by manual human annotation or automatically.

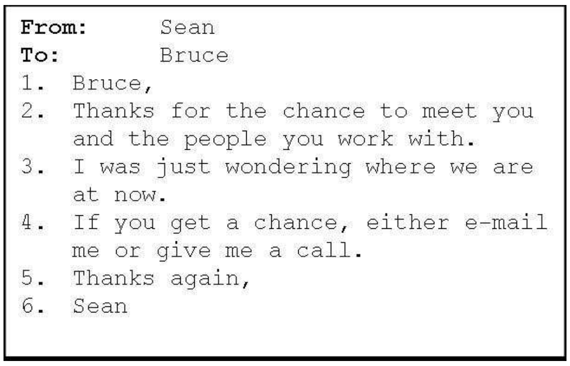
We apply some guiding principles in attempting to resolve cases of ambiguity and controversy. Where appropriate, we prioritise recall, based on the intuition that for our chosen application of highlighting requests and commitments in a message, missing tasks will be more problematic than mistakenly identifying non-tasks.

4.1 Locus Ambiguity

As indicated in Table 1, our experiments show that annotating at the message level achieves higher interannotator agreement than at the sentence level. One contributing factor to this difference stems from ambiguity around the actual location of a request or commitment in the text of an email message. We refer to this problem as ambiguity about the *locus* of requests and commitments.

Figure 1 shows an example of an email with ambiguous request locus.[2] At the message level, our annotators agree that there is a request for Bruce to contact Sean. When asked to find requests at the sentence level, however, the presence of both an indirect and a direct realisation of the same underlying request leads to ambiguity about the locus: all annotators agreed that Sentence 4 is a request, but Sentence 3 was controversial. After discussion, all

[2]The numbers here have been added to aid in our discussion of the example and are not part of the message as annotated.



```
From:     Sean
To:       Bruce
1.  Bruce,
2.  Thanks for the chance to meet you
    and the people you work with.
3.  I was just wondering where we are
    at now.
4.  If you get a chance, either e-mail
    me or give me a call.
5.  Thanks again,
6.  Sean
```

Figure 1: Ambiguous Locus

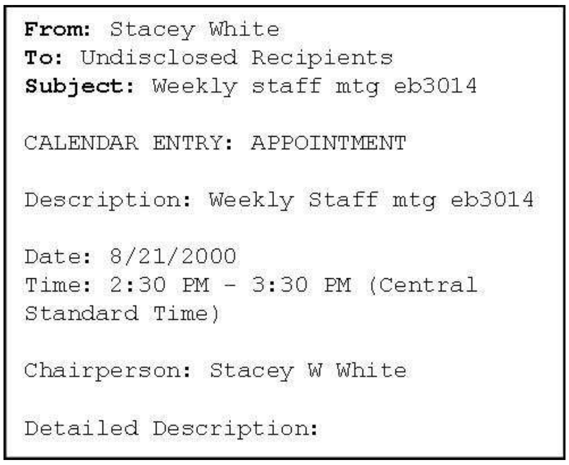
annotators agreed that they would mark Sentence 3 as an (indirect) request *if Sentence 4 was not present*. With both sentences present, however, annotators disagreed about whether Sentence 3 is a request or provides background information. In our sentence-level experiments, we fixed the unit of annotation to be a single sentence. Annotators were thus required to mark Sentences 3 and 4 independently; they could not group sentences together as a single request spanning multiple sentences. Annotators had to choose between marking only the direct realisation (Sentence 4) and marking both sentences as requests.

Our guidance for dealing with cases of locus ambiguity is: **each sentence realising a request or commitment, whether directly or indirectly, should be marked**. This is consistent with our guiding philosophy of prioritising recall, as noted earlier. Thus, in Figure 1, both Sentences 3 and 4 are requests.

The email in Figure 2 offers an example of a meeting request with a different type of locus ambiguity. While it seems clear that the email represents a request at the message level, it is unclear which utterance(s) in the email, if any, should be marked as the locus of the request. If we consider the message also to represent a commitment on Stacey's part to attend, the same problem occurs for identifying the locus of the commitment.

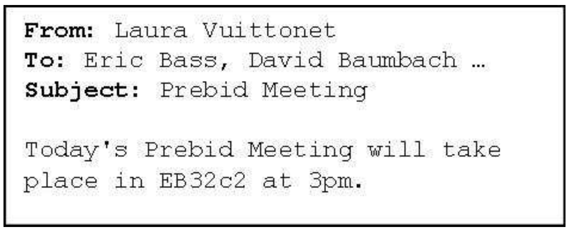
As illustrated in Figures 1 and 2, locus ambiguity is abstracted away when annotating at the message level. This explains the higher interannotator agreement shown in Table 1.

In future work, we plan to explore further the na-



```
From: Stacey White
To: Undisclosed Recipients
Subject: Weekly staff mtg eb3014

CALENDAR ENTRY: APPOINTMENT

Description: Weekly Staff mtg eb3014

Date: 8/21/2000
Time: 2:30 PM - 3:30 PM (Central
Standard Time)

Chairperson: Stacey W White

Detailed Description:
```

Figure 2: Locus ambiguity for a Meeting Request



```
From: Laura Vuittonet
To: Eric Bass, David Baumbach …
Subject: Prebid Meeting

Today's Prebid Meeting will take
place in EB32c2 at 3pm.
```

Figure 3: A Meeting Announcement Email

ture of locus ambiguity to determine its importance amongst the other phenomena presented in this paper, and to better understand different patterns of locus ambiguity (for example, contiguous indirect–direct realisations and non-contiguous realisations).

4.2 Meetings

Meeting requests and announcements are very common in the workplace email we analysed (both our own inboxes and the Enron corpus). Our current annotation guidelines provide the following principle: **all meeting announcements are requests, in the sense that they are usually implicit requests to attend**.

Following this principle, there are requests in the messages of both Figures 2 and 3. The implicit request in Figure 3 is, however, more readily identified at the sentence level. Structured messages used for exchanging calendar data via email (e.g., iCalendar or vCalendar messages) are also considered to be requests.

A related issue is whether the sender is committing to attend the meeting. This largely depends on whether the meeting is an activity in which both sender and recipient are involved, and whether the sender is interpreted to be committing to attending or otherwise acting in relation to the meeting. This can only be resolved using the available context.

An alternative approach, employed by other researchers such as Corston-Oliver et al. (2004), is to deal with this ambiguity by creating a separate *meeting* category, distinct from the request category. This, however, introduces the problem of cases that straddle the boundaries between the categories.

4.3 Pleasantries

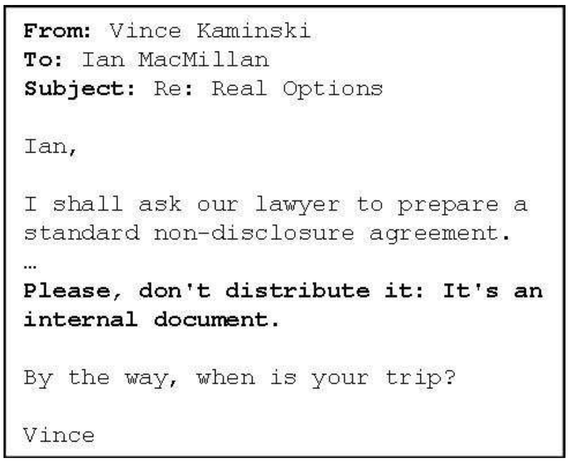
Utterances like *Let me know if you have any questions* are extremely common in workplace email. Corston-Oliver et al. (2004) made a similar observation, labelling such utterances "formulaic endings". In some cases, these utterances actually carry the illocutionary force of a request for the recipient to act and/or a commitment from the speaker. In many cases, however, their presence is mostly a matter of conformance with social norms. Where no obligation is assigned, we call these *pleasantries*.

Pleasantries resemble the surface form of requests or commitments, but place very little obligation, or no obligation at all, on the recipient or other identified agent to act or respond. Correspondingly, we have our third principle: **pleasantries are not requests or commitments**.

Like Corston-Oliver et al., we believe the best approach to distinguishing between pleasantries and actual requests and commitments is to consider the context of the entire message. Annotators are instructed to use their judgement to distinguish when utterances such as *Let me know if you have any questions* should be interpreted as mere social convention and when they represent requests for review and comment and/or an offer of future assistance from the sender.

Even with explicit guidance to consider the entire context of a message, there remain cases of unresolvable disagreement between our annotators around the interpretation of specific utterances. Ultimately, the decision is subjective. Automated tools thus need to adapt to the preferences of a specific user and the norms for communication between different email interlocutors.

In our current annotation guidelines, we have cre-



From: Vince Kaminski
To: Ian MacMillan
Subject: Re: Real Options

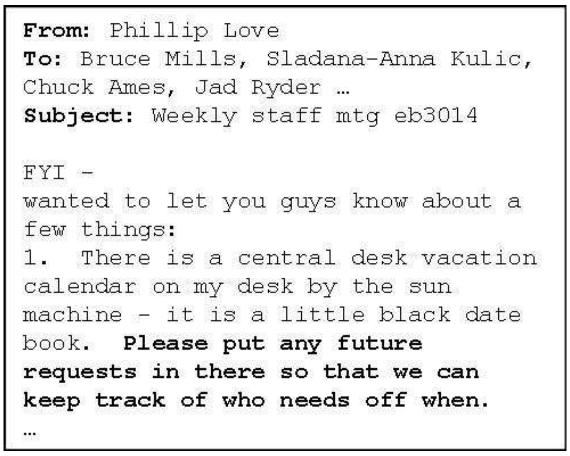
Ian,

I shall ask our lawyer to prepare a standard non-disclosure agreement.
…
Please, don't distribute it: It's an internal document.

By the way, when is your trip?

Vince

Figure 4: Request for Inaction



From: Phillip Love
To: Bruce Mills, Sladana-Anna Kulic, Chuck Ames, Jad Ryder …
Subject: Weekly staff mtg eb3014

FYI -
wanted to let you guys know about a few things:
1. There is a central desk vacation calendar on my desk by the sun machine - it is a little black date book. **Please put any future requests in there so that we can keep track of who needs off when.**
…

Figure 5: Process Instructions

ated a separate category for pleasantries to help quantify the disagreement that arises from this phenomenon; this will support more detailed analysis of the nature of pleasantries.

4.4 Requests for Inaction

Requests for inaction prohibit action or request negated action. They are sometimes called *prohibitives* (Sadock and Zwicky, 1985). An example (with our emphasis added, here and in subsequent examples) is shown in Figure 4.

By definition, requests for inaction do not require action, so one would not expect such utterances to require a new entry in the recipient's task list. As a result, definitions based on the suitability of an action for inclusion in a recipient's task list would ignore requests for inaction. Clearly, however, such requests place an obligation on the recipient, thus our fourth principle: **requests for inaction are considered to be requests**.[3]

The use of negation illustrates again the complex relationship between the surface text and underlying speech act. Utterances that request action using a negated surface form, such as *Don't forget to send me your comments* (an alternate realisation of the utterance *Please send me your comments*), are requests for action, not inaction.

[3]Note that our inclusion of requests for inaction as requests differs from our original stance taken in (Lampert et al., 2008).

4.5 Process Instructions

Another class of edge case requests stems from email messages that contain process instructions or hypothetical requests or commitments: instructions of the kind that one might 'file for later use'. An example is shown in Figure 5.

In our discussions over disagreements, one property played a major role in whether an instruction should be marked as a request: the likelihood of the situation leading to the actual execution of the described action. An utterance such as *In the event of a fire, please leave quickly and calmly via the closest emergency exit* is an example of a low-probability-action that our annotators were not likely to mark as a request.

After careful consideration, our current annotations guidelines **leave the decision about specific instances of process instructions to annotator judgement based on the local context**. We expect that analysis of more annotated data will give us a better empirical understanding of the nature of process instructions, and the distinction between instructions that function as requests and are relevant to our proposed email applications, and those that are not.

4.6 Attachment Review Requests

A common use of email is to disseminate documents as attachments. Attachments are commonly, though certainly not always, accompanied by an utterance along the lines of *Please see attached*. In keeping with our goal to include any type of action as a re-

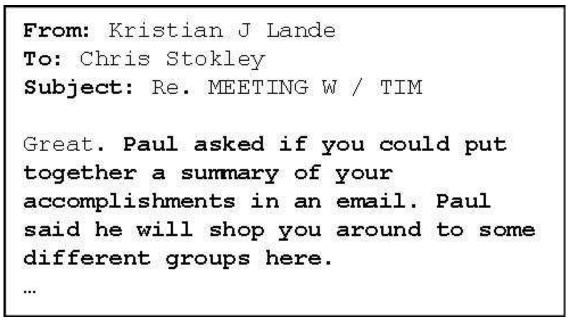

```
From: Kristian J Lande
To: Chris Stokley
Subject: Re. MEETING W / TIM

Great. **Paul asked if you could put
together a summary of your
accomplishments in an email. Paul
said he will shop you around to some
different groups here.**
...
```

Figure 6: Reported Request and Commitment

quest, we consider such utterances to be requests to read, archive, or otherwise act on the attached document.

How should we treat attachments where no utterance in the email body directs or requests the recipient to do anything with the attachment? One possible interpretation is that every attachment carries with it an implicit request to read it or otherwise act on its content. The same, however, could be argued for any message that arrives in a user's inbox, resulting in every message being considered to convey an implicit request to be read. Such an interpretation seems unintuitive and inconsistent with our focus on linguistic email content. Thus, our sixth principle is to **only consider as a request attachments accompanied by a textual request (whether implicit or explicit)**.

4.7 Reported Requests and Commitments

Reported requests and commitments are another complication in real-world email. An example email containing both a reported request and a reported commitment is shown in Figure 6. In this case, both are likely to place an obligation: on the recipient (from the request) and on Paul (from the commitment).

Some reported commitments or requests do not actually place an obligation on any agent. An example is shown in Figure 7.

We have found it difficult to find reliable distinctions between reported requests and commitments that should and should not be marked as actual requests and commitments. Consequently, our principle is to **use the local context to determine the function of a reported request or commitment**.

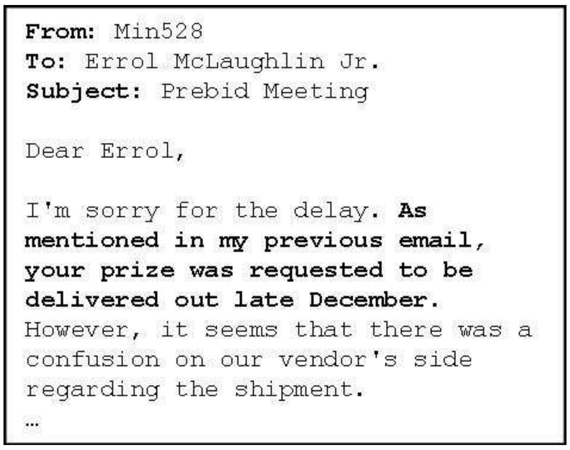
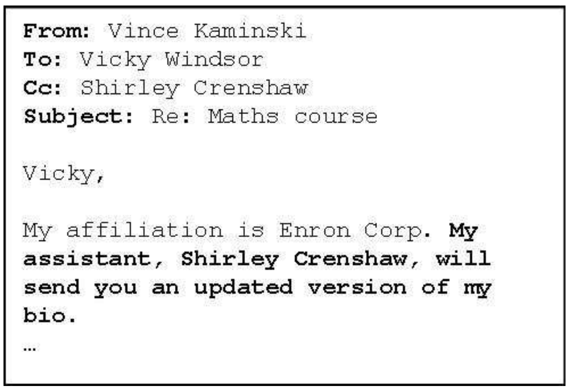

```
From: Min528
To: Errol McLaughlin Jr.
Subject: Prebid Meeting

Dear Errol,

I'm sorry for the delay. **As
mentioned in my previous email,
your prize was requested to be
delivered out late December.**
However, it seems that there was a
confusion on our vendor's side
regarding the shipment.
...
```

Figure 7: Reported Request that is not a Request

4.8 Third-Party Commitments as Requests

Some commitments from the sender place an obligation on one of the recipients (usually one who is cc'ed). According to our definitions, such commitments also function as requests. Consider, for example, the utterance: *My assistant, Shirley Crenshaw, will send you an updated version of my bio* in Figure 8. When Shirley is a recipient of the email, this commitment also functions as a request to Shirley. In contrast, the same email sent without Shirley as a recipient does not function as a request, since it is not communicated to Shirley. It does still function as a commitment for a third party. Our annotation principle is to **consider any commitment that places an obligation on a recipient of the message as both a request and a commitment**.

Commitments functioning as requests are found in both the Enron corpus and in other workplace email we have analysed. They are typically from managers to their personal assistants or direct reports, or sometimes between peers. Because it is not possible in the case of the Enron corpus to know who receives email sent to group aliases or mailing lists, we only consider an individual's email address in determining whether the committed person is a recipient of the email. This issue would not arise, of course, when annotating from the perspective of a specific user's inbox with the aim of identifying obligations for that user.



```
From: Vince Kaminski
To: Vicky Windsor
Cc: Shirley Crenshaw
Subject: Re: Maths course

Vicky,

My affiliation is Enron Corp. My
assistant, Shirley Crenshaw, will
send you an updated version of my
bio.
...
```

Figure 8: Third Party Commitment as a Request

5 Conclusions

In this paper, we have identified eight phenomena in email messages that make the identification of requests and commitments difficult. We believe that any work that attempts to process requests and commitments in email must acknowledge the existence of these phenomena. For each phenomenon, we have provided a definition and indicated how we approach it in our annotation, summarised as follows:

Locus Ambiguity: Each and every sentence which realises a request or commitment, whether directly or indirectly, should be marked, even if this means that two or more realisations of the same request or commitment are marked.

Meetings: All meeting announcements are considered to be requests, in the sense that they are usually implicit requests to attend.

Pleasantries: Pleasantries are not considered to be requests or commitments.

Requests for Inaction: Requests for inaction are considered to be requests.

Process Instructions: Whether or not a specific instance of a process instruction constitutes a request depends on the context and is left to the annotator's judgement.

Attachment Review Requests: Attachments are only considered to constitute requests when they are accompanied by text in the message that can be interpreted, implicitly or explicitly, as a request.

Reported Requests and Commitments: Whether or not a specific instance of a reported request or commitment constitues an actual request or commitment depends on the context and is left to the annotator's judgement.

Third-Party Commitments as Requests: Any commitment that places an obligation on a recipient of the message is considered to be both a request and a commitment.

While others may want to deal with these categories of phenomena differently, the categories themselves cannot be ignored. As we have tried to demonstrate in this paper, we cannot rely on simple canonical examples of requests and commitments as being representative of the ways in which these speech acts manifest themselves.

The implications of these edge cases for automating the detection of requests and commitments in email are numerous. Many of the phenomena identified are not easily detectable from the surface form of text in an email, often requiring access to contextual information. This context must thus be modelled in the features used when applying statistical machine learning techniques to the problem. We recognise that some categories will still remain ambiguous because they require context that cannot be easily modelled; in such cases, we would expect a tool to resort to a human-in-the-loop mode of operation.

A more significant issue is the evaluation of automatic request and commitment classifiers, given the challenge of reaching reliable human agreement. It is not possible to measure performance against an objective gold standard. Evaluating in the context of real users' mailboxes and task contexts is likely to be more relevant and meaningful than assessing against an abstract gold standard. This, of course, comes with huge challenges around privacy, access to data and experimental repeatability.

A collection of email data, independently labelled by five annotators according to the guidelines described in this paper, is available for download from `http://annotate.thoughtlets.org`.

References

Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. 2003. Taking email to task: The design and evaluation of a task management centred email tool. In *Computer Human Interaction Conference*, CHI, pages 345–352, Ft Lauderdale, Florida, USA, April 5-10.

Beatrice M. Camino, Allen E. Milewski, David R. Millen, and Thomas M. Smith. 1998. Replying to email with structured responses. *International Journal of Human-Computer Studies*, 48(6):763–776, June.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In Dekang Lin and Dekai Wu, editors, *Conference on Empirical Methods in Natural Language Processing*, pages 309–316, Barcelona, Spain.

Simon H. Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *ACL-04 Workshop: Text Summarization Branches Out*, pages 43–50, July.

Nicolas Ducheneaut and Victoria Bellotti. 2001. E-mail as habitat: an exploration of embedded personal information management. *Interactions*, 8(5):30–38, September/October.

Jade Goldstein and Roberta Evans Sabin. 2006. Using speech acts to categorize email and identify email genres. In *Proceedings of the 39th Hawaii International Conference on System Sciences*, page 50b.

Hamid Khosravi and Yorick Wilks. 1999. Routing email automatically by purpose not topic. *Journal of Natural Language Engineering*, 5:237–250. Cambridge University Press.

William Labov and David Fanshel. 1977. *Therapeutic Discourse: Psychotheray as Conversation*. Academic Press, New York.

Andrew Lampert, Cécile Paris, and Robert Dale. 2007. Can requests-for-action and commitments-to-act be reliably identified in email messages? In *Proceedings of the 12th Australasian Document Computing Symposium*, pages 48–55, Melbourne, Australia, December 10.

Andrew Lampert, Robert Dale, and Cécile Paris. 2008. The nature of requests and commitments in email messages. In *Proceedings of the AAAI Workshop on Enhanced Messaging*, pages 42–47, Chicago, USA, July 13. AAAI Press. Technical Report WS-08-04.

Anton Leuski. 2004. Email is a stage: discovering people roles from email archives. In *Proceedings of Annual ACM Conference on Research and Development in Information Retrieval*, Sheffield, UK.

Wendy E. Mackay. 1988. More than just a communication system: Diversity in the use of electronic mail. In *ACM conference on Computer-supported cooperative work*, pages 344–353, Portland, Oregon, USA. MIT, ACM Press.

Denise E. Murray. 1991. *Conversation for Action: The Computer Terminal As Medium of Communication*. John Benjamins Publishing Co.

Jerry M. Sadock and Arnold Zwicky, 1985. *Language Typology and Syntactic Description. Vol.1 Clause Structure*, chapter Speech act distinctions in syntax, pages 155–96. Cambridge University Press.

Simon Scerri, Myriam Mencke, Brian David, and Siegfried Handschuh. 2008. Evaluating the ontology powering smail — a conceptual framework for semantic email. In *Proceedings of the 6th International conference of Language Resources and Evaluation*, Marrakech, Morocco, May 28-30.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

John Sinclair and Richard Malcolm Coulthard. 1975. *Towards and Analysis of Discourse - The English used by Teachers and Pupils*. Oxford University Press.

Steve Whittaker and Candace Sidner. 1996. Email overload: exploring personal information management of email. In *ACM Computer Human Interaction conference*, pages 276–283. ACM Press.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 73-78

Text Mining Based Query Expansion for Chinese IR

Zhihan Li Yue Xu Shlomo Geva

Faculty of Information Technology
Queensland University of Technology
Brisbane, QLD 4001, Australia
Z7.li@student.qut.edu.au {yue.xu,s.geva}@qut.edu.au

Abstract

Query expansion has long been suggested as a technique for dealing with word mismatch problem in information retrieval. In this paper, we describe a novel query expansion method which incorporates text mining techniques into query expansion for improving Chinese information retrieval performance. Unlike most of the existing query expansion strategies which generally select indexing terms from the top N retrieved documents and use them to expand the query, in our proposed method, we apply text mining techniques to find patterns from the retrieved documents which contain relevant terms to the query terms, then use these relevant terms which can be indexing terms or indexing term patterns to expand the query. The experiment with NTCIR-5 collection shows apparent improvement in both precision and recall.

1 Introduction

The amazing growing speed in the number of Chinese Internet users indicates that building Chinese information retrieval systems is in great demand. This paper presents a method aimed at improving the performance of Chinese document retrieval.

Unlike English text in which sentences are sequences of words delimited by white spaces, in Chinese text, sentences are represented as strings of Chinese characters without separating spaces between words. For Chinese information retrieval, the query is usually a set of Chinese words rather than a sequence of Chinese characters. For character based Chinese information retrieval, since the texts are not segmented, the retrieved documents which contain the character sequence of the query may not be relevant to the query since they may not contain the words in the query. Therefore, the quality of character based Chinese information retrieval is not satisfactory. On the other hand, a study has shown that the relationship between segmentation and retrieval is in fact non-monotonic, that is, high precision of segmentation alone may not improve retrieval performance (Peng, Huang, Schuurmans and Cercone 2002). In this paper we propose a Chinese information retrieval model which combines character based retrieval and word based ranking to achieve better performance.

Another critical problem is that the original query may not represent adequately what a user wants. By appending additional terms to the original query, Query Expansion attempts to construct a richer expression to better represent the user's information need. Pseudo relevance feedback is a popular technique for query expansion. The basic idea is to automatically extract additional relevant terms from the top ranked documents in the initial result list. These terms are added to the original query, and the extended query is executed with the expectation of improved performance. In this paper, we propose a new approach to improving the performance of Chinese document retrieval by expanding queries with highly correlated segmented words, generated by using text mining techniques.

The remainder of this paper is structured as follow. Section 2 briefly reviews some related work. In Section 3, we describe our retrieval model, and then present the text mining based query expansion method in Section 4. Section 5 presents experimental results and Section 6 concludes the paper.

2 Related Work

Unlike English text in which sentences consist of words delimited by white spaces, in Chinese text, sentences are represented as strings of Chinese characters without delimiters. Therefore, Chinese word segmentation is the first phase in Chinese language processing and has been widely studied for many years (Gao and Li 2005; Xue 2003; Sproat and Shih 2002; Wang, Liu and Qin 2006). Both Chinese characters and words can be used as the indexing units for Chinese IR. Several approaches have shown that single character indexing can produce good results, but word and bi-gram indexing can achieve slightly better performance. This however incurs greater time and space complexity with limited performance improvement (Sproat and Shih 2002; Li 1999; Kwok 1997; Peng, Huang, Schuurmans and Cercone 2002). In this paper, we propose a ranking method that combines character indexing and segmented word indexing to re-rank retrieved documents and promote relevant documents to higher positions.

Pseudo-relevance feedback is an important query expansion technique for improving IR performance (Qiu and Frei 1993; Sun, Ong and Chua 2006; Robertson and Jones 1976). The basic insight which motivates pseudo relevance feedback is that often the top of the initially ranked list of results contains a relatively high proportion of relevant documents. The conjecture is that despite the presence of some irrelevant documents, these retrieved documents might still be used to identify relevant terms that co-occur in the relevant documents. These terms are then used to modify the original query and better reflect the user's information needs. With the expanded query, a second retrieval round is performed and the returned result is expected to contain more relevant documents which have been missed in the first retrieval round. For pseudo relevance feedback query expansion, the most important task is to find the terms from the retrieved documents that are considered relevant to the query. Therefore, relevant term selection is crucial in pseudo relevance feedback query expansion. The standard criteria for selecting relevant terms have been proposed using tf/idf in vector space model (Rocchio 1997) and probabilistic model (Robertson and Jones 1976). Query length has been considered in (Kwok, Grunfeld and Chan 2000) for weighting expansion terms and some linguistic features also have been tried in (Smeaton and Rijsbergen 1983). We are proposing to use text mining techniques to find the relevant terms.

Data Mining is about analyzing data and finding hidden patterns using automatic or semi-automatic means. Text mining is a research field of data mining which refers to the process of deriving high quality patterns and trends from text. We are proposing to apply text mining techniques to finding frequent patterns in the retrieved documents in the first retrieval round which contain query terms. These patterns provide us with the candidate sequences to find more terms which are relevant to the original query. The application of text mining to information retrieval may improve precision and recall.

3 Retrieval Model

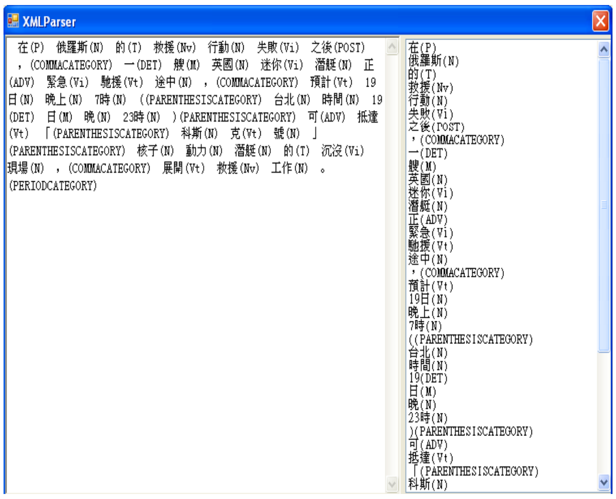
In general, character indexing based IR can retrieve most of the relevant documents as long as they contain the query terms (the query terms are sequences of Chinese characters but not necessarily sequences of segmented words in the retrieved documents since the documents are not segmented). However, the retrieval performance is not necessarily good. This is because many irrelevant documents are highly ranked due to high query term frequency corresponding to instances of the query term sequences which are actually not valid words but rather correspond to what would be incorrect word segmentation. On the other hand, the word indexing based IR can apply better ranking and therefore achieve somewhat better performance than that of character indexing based IR. The improvement is limited since some relevant documents may not contain the query terms as segmented words and thus won't be retrieved. In this section, we describe our approach to Chinese information retrieval in which, we firstly create two indexing tables from the data collection, a Chinese character indexing table and a segmented Chinese word indexing table; secondly, the relevant documents are retrieved based on the character indexing, then the retrieved documents are ranked by a method proposed in this paper that ranks the retrieved documents based on both character indexing and word indexing.

3.1 Indexing and Retrieval Model

The first task is word segmentation. We used an online program provided by the Institute of Information science, Academia Sinica in TaiWan to segment the documents. The segmentation precision of the system is approximately 95% as reported in (Ma and Chen 2002), and most importantly, this system not only accomplishes word segmentation, but also incorporates POS (part of speech) annotation information into the segmented documents which is very important for this research since we are therefore able to utilize the POS information to improve the efficiency and effectiveness of Chinese IR based on word indexing. For example, for the following original Chinese text:

在俄羅斯的救援行動失敗之後，一艘英國迷你潛艇正緊急馳援途中，預計 19 日晚上 7 時台北時間 19 日晚 23 時可抵達科斯克號核子動力潛艇的沉沒現場，展開救援工作。

We can get the following segmented text:



```
XMLParser
 在(P)  俄羅斯(N)  的(T)  救援(Nv)  行動(N)  失敗(Vi)  之後(POST)
 ，(COMMACATEGORY)  一(DET)  艘(M)  英國(N)  迷你(Vi)  潛艇(N)  正
(ADV)  緊急(Vi)  馳援(Vt)  途中(N)  ，(COMMACATEGORY)  預計(Vt)  19
日(N)  晚上(N)  7時(N)  ((PARENTHESISCATEGORY)  台北(N)  時間(N)  19
(DET)  日(M)  晚(N)  23時(N)  )(PARENTHESISCATEGORY)  可(ADV)  抵達
(Vt)  「(PARENTHESISCATEGORY)  科斯(N)  克(Vt)  號(N)  」
(PARENTHESISCATEGORY)  核子(N)  動力(N)  潛艇(N)  的(T)  沉沒(Vi)
現場(N)  ，(COMMACATEGORY)  展開(Vt)  救援(Nv)  工作(N)  。
(PERIODCATEGORY)
```

```
在(P)
俄羅斯(N)
的(T)
救援(Nv)
行動(N)
失敗(Vi)
之後(POST)
，(COMMACATEGORY)
一(DET)
艘(M)
英國(N)
迷你(Vi)
潛艇(N)
正(ADV)
緊急(Vi)
馳援(Vt)
途中(N)
，(COMMACATEGORY)
預計(Vt)
19日(N)
晚上(N)
7時(N)
((PARENTHESISCATEGORY)
台北(N)
時間(N)
19(DET)
日(M)
晚(N)
23時(N)
)(PARENTHESISCATEGORY)
可(ADV)
抵達(Vt)
「(PARENTHESISCATEGORY)
科斯(N)
```

Figure 1. Chinese Word Segmentation Example

The left part of figure 1 is the segmented document and the right part lists all words in the segmented document and each word is associated with a POS tag immediately after the word.

We built two indexing tables, one is character based and the other is segmented word based. In earlier research (Lu, Xu and Geva 2007), we have developed a character based indexing system to perform character based retrieval and ranking. In this paper, we directly use the existing retrieval model, which is the traditional Boolean model, a simple model based on set theory and Boolean algebra, to perform document retrieval.

3.2 Ranking Method

In our previous research, we used the following ranking model to calculate the ranking value of a retrieved document to determine the top N relevant documents:

$$d_{rank} = n^5 \sum_{i=1}^{m} tf_i^c \times idf_i^c \tag{1}$$

Here, *m* is the number of query terms, *n* is the number of distinct query terms that appear in the document as character sequences (not necessarily segmented words). tf_i^c is the frequency of the *i*th term in the document and idf_i^c is the inverse document frequency of the *i*th term in the collection. The equation can ensure two things: firstly, the more distinct query terms are matched in a document, the higher the rank of the document. For example, a document that contains four distinct query terms will almost always have higher rank than a document that contains three distinct query terms, regardless of the query terms frequency in the document. Secondly, when documents contain a similar number of distinct terms, the score of a document will be determined by the sum of query terms' *tf-idf* value, as in traditional information retrieval.

In this paper, we use the following equation to calculate documents' ranking scores, which is simply the average of character based ranking and word based ranking:

$$d_{rank}^{c+w} = \frac{n_c^5 \sum_{i=1}^{m} tf_i^c \times idf_i^c + n_w^5 \sum_{k=1}^{m} tf_i^w \times idf_i^w}{2} \tag{2}$$

Where n_c is the number of distinct query terms which appear in the document as character sequences (not necessarily segmented words). tf_i^c and idf_i^c are the same as in Equation (1), tf_i^w is the frequency of the *i*th term in the document as a segmented word, and idf_i^w is the inverse document frequency of the *i*th term in the collection as a segmented word, and n_w is the number of query terms which appear in the document as a segmented word.

4 Query Expansion

It has been recognized that user queries consisting of a few terms are inadequate to fully reflect users' information needs and lead to a poor coverage of the relevant documents. Query expansion is the technique widely used to deal with this problem. In this section, we describe a new method that applies text mining techniques to find terms from the retrieved documents that are highly correlated to the query, and then performs a second retrieval using the expanded query with these relevant terms. In the first part of this section, we introduce the method to generate a set of candidate relevant terms from the retrieved documents. Then, in the second part of this section, we introduce a method to select the most relevant terms to expand the original query.

In this stage, the top N retrieved documents from the first retrieval round are converted to a set of transactions which are used to mine frequent patterns using text mining methods such as FP-Tree method (Han, Pei and Yin 2000; Zou, Chu, Johnson and Chiu 2001; Agrawal and Srikant 1994). Query terms are usually nouns. So, it is reasonable to only extract patterns of nouns rather than patterns of all words in the retrieved documents. Therefore, from the retrieved documents we first eliminate all non-noun words based on POS tag information, then construct a collection of transactions each of which consists of the nouns in a sentence from a retrieved document. Thus, all sentences in the retrieved documents are included in the transaction collection if they contain nouns. A sentence will be excluded from the transaction collection if it does not contain nouns. For example, we extracted 18 unique nouns from the example in Figure 1. The 18 nouns form a vector with size 18:

俄羅斯(Russia), 行動 (Action), 英國 (England), 潛艇 (Submarine), 途中 (Way), 19 日 (19th), 晚上 (Evening), 7 時 (7PM), 台北 (TaiBei), 時間 (Time), 晚 (Night), 23 時 (23PM), 科斯 (Kesi), 號 (Name), 核子 (Nucleon), 動力 (Power), 現場 (Scene), 工作(Work).

Each transaction created from one sentence in a retrieved document is a vector of 1s and 0s, each element of the transaction corresponds to a noun with value 1 indicating the corresponding noun appearing in the sentence and otherwise 0. The number of transactions is the number of sentences in the retrieved top N documents. The size of the transaction is the number of unique nouns in the top N documents.

The collection generated from the example is shown in Table 1:

sentence	*Transaction*
在俄羅斯的救援行動失敗之後	110000000000000000
一艘英國迷你潛艇正緊急馳援途中	001110000000000000
預計 19 日晚上 7 時台北時間 19 日晚 23 時可抵達科斯克號核子動力潛艇的沉沒現場	000101111111111110
展開救援工作	000000000000000001

Table 1. Example collection of transactions

Any pattern mining algorithm can be used to generate frequent patterns from the constructed collection mentioned above. In this paper, we choose the popular used FP-Tree algorithm to perform the pattern mining task. Let Q={q_1, …, q_m} be the query containing m query terms and $FP = \{p_1, p_2, \cdots, p_n\}$ be the set of mined frequent patterns, *Supp*(p_k) be the support value of pattern p_k. For a query term q_i, the set of patterns which contain the query term is denoted as $FP_i = \{p_k | p_k \in FP, q_i \in p_k\}$. For a pattern p_j and a query term q_i, the set of patterns which contain both p_j and q_i is denoted as $FP_{ij} = \{p_k | p_k \in FP, q_i \in p_k, p_j \in p_k\}$, and the set of patterns which contain p_j is denoted as $FP_j = \{p_k | p_k \in FP, p_j \in p_k\}$.

The following equation is proposed to calculate the relevancy between a query term q_i and a pattern p_j which is denoted as $R_{(q_i, p_j)}$:

$$R_{(q_i,p_j)} = W_1 \frac{N_{ij}}{N_i} + W_2 \frac{\sum_{p_k \in FP_{ij}} Supp(p_k)}{N_{ij}} + W_3 \frac{\sum_{p_k \in FP_j} Supp(p_k)}{N_j} \tag{3}$$

Where $N_{ij} = |FP_{ij}|$, $N_i = |FP_i|$, $N_j = |FP_j|$, which are the number of frequent patterns. The first part in Equation (3) measures the ratio between the patterns which contain both the query term q_i and the pattern p_j. The higher the ratio is, the more the pattern p_j is relevant to the query q_i. The second part in the equation is the average support value of the patterns containing both the query term q_i and the pattern p_j. A high average support indicates that the co-occurrence of the query term q_i and the pattern p_j is high which in turn indicates that q_i and p_j are highly correlated with each other. Similarly, the third part is the average support value of the patterns containing only the pattern p_j. A high average support of pattern p_j means that p_j is a popular pattern in the retrieved documents and may have a high relevance to the query term. Equation (3) is a linear combination of the three parts with W_1 , W_2 and W_3 as coefficients which can be controlled by users to adjust the weights of the three parts.

For the whole query, the relevancy value R_{p_j} of pattern p_j is calculated as follows:

$$R_{p_j} = 100 \times \sum_{q_i \in Q} R_{(q_i,p_j)} \tag{4}$$

All these patterns are ranked by relevancy values. Based on the relevancy value R_{p_j}, we select the top 3 patterns and use the words in the patterns to expand the original query.

5 Experimental Results

We select NTCIR-5 Chinese corpus as our dataset, which includes 434,882 documents in traditional Chinese. We built two indexing tables, one contain indexes of all Chinese characters appearing in the documents, and the other contains all Chinese words with POS information produced by the online segmentation program developed by the Institute of Information science, Academia Sinica in TaiWan. Fifty queries are used and each query is a simple description which consists of one or several terms. We use the average precision (denoted as P in Table 2) and average recall (denoted as R) of the top 10, 15, 20, 30 and 100 retrieved documents to evaluate the performance of the proposed Chinese IR model with the text mining based query expansion (denoted as QE(TM)) by comparing with the character based model without query expansion (denoted as C) , the word-character based model without query expansion (denoted as W-C), and the proposed Chinese IR model with the popularly used standard query expansion method Rocchio (denoted as QE(R)). In the experiments, W1, W2, W3 are set to 0.8, 0.18, 0.02, respectively. Additionally, we set the support value as 0.1 for using FP-Tree method to mine frequent patterns. The experiment results are given in Table 2.

TOP N	**C**		**W-C**		**QE (R)**		**QE(TM)**	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
10	39.7	19.6	39.9	20.1	41.6	23.2	49.2	26.8
15	35.2	26.1	35.5	26.5	39.7	28.9	44.1	34.0
20	32.7	30.9	32.9	31.3	35.9	32.1	39.2	38.7
30	27.5	36.7	27.5	35.7	30.2	38.8	33.0	44.5
100	14.1	53.8	14.6	53.9	15.1	60.4	16.3	64.7
Ave	29.9	33.4	30.1	33.5	32.5	36.8	36.4	41.7

Table 2 Precision and Recall

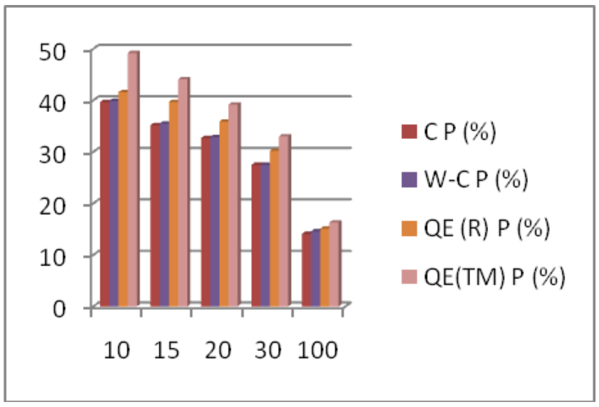


Figure 2 Precision

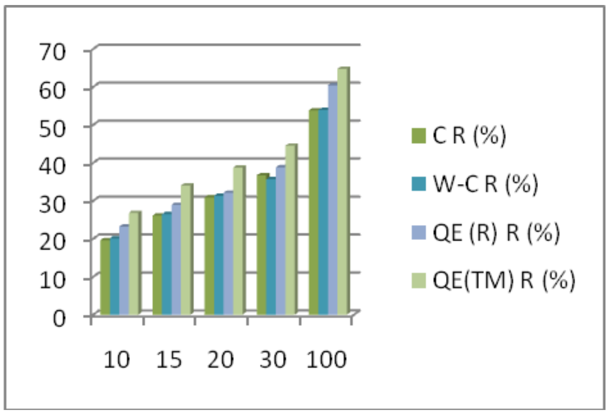


Figure 3 Recall

Table 2 shows the precision and recall of the four different retrieval approaches. From Table 2, we can see that the performance is improved slightly from the character based model to the word-character based model, the precision is improved by 0.2% on average from 29.9% to 30.1% and the recall is improved by 0.1% on average from 33.4% to 33.5%. With the Rocchio standard query expansion, we achieved a little more improvement, the precision is improved by 1.4% on average from 30.1% to 32.5% and the recall is improved by 3.3% from 33.5% to36.8%. However, with our text mining based query expansion method, we achieved much larger improvements; precision is improved by 6.3% on average from 30.1% to 36.4% and the recall is improved by 8.2% from 33.5% to 41.7%.

6 Conclusion

In this paper, we proposed an approach to improve the performance of Chinese information retrieval. The approach includes two aspects, retrieval based on segmented words and text query expansion using text mining techniques. The experiment results show that Chinese word segmentation does not bring significant improvement to Chinese information retrieval. However, the proposed text mining based query expansion method can effectively improve the performance of the Chinese IR and the improvement is much greater than that achieved by using the standard query expansion method Rocchio.

References

Chengye Lu, Yue Xu, Shlomo Geva. 2007. *Translation disambiguation in web-based translation extraction for English-Chinese CLIR*. Proceedings of the 2007 ACM symposium on Applied computing, 819-823.

F. Peng, X. Huang, D. Schuurmans, and N. Cercone. 2002. *Investigating the relationship between word segmentation performance and retrieval performance in Chinese IR*, Proceedings of the 19th international conference on Computational linguistics, 1-7.

J. Han, J. Pei and Y. Yin. 2000. *Mining Frequent Patterns without Candidate Generation*. Proc of the 2000 ACM International Conference on Management of Data, Dallas, TX, 2000, 3-12.

Jianfeng Gao and Mu Li. 2005. *Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach*. Computational Linguistics, MIT. 531-574, Vol 31, Issue 4.

K. L. Kwok. 1997. *Comparing representations in Chinese information retrieval*. Proc. Of the ACM SIGIR97, 34-41.

Kwok, K.L, Grunfeld, L., Chan, K. 2000. *THREC-8 ad-hoc, query and filtering track experiments using PIRCS*, In TREC10.

Nianwen Xue. 2003. *Chinese Word Segmentation as Character Tagging*. Computational Linguistics and Chinese Language Processing, Vol 8, No 1, Pages 29-48.

P. Li. 1999. *Research on improvement of single Chinese character indexing method*. Journal of the China Society for Scientific and Technical Information, 18(5).

Q. Zou, W. Chu, D. Johnson and H. Chiu. 2001. *A Pattern Decomposition (PD) Algorithm for Finding all Frequent Patterns in Large Datasets.* Proc of the 2001 IEEE International Conference on Data Mining (ICDM01), San Jose, California, 673-674.

Qiu Y. and Frei H. 1993. *Concept based query expansion*. In Proceedings of SIGIR 1993, pp. 160-169.

R. Agrawal and R. Srikant. 1994. *Fast algorithms for mining association rules*. Proc. Of the 1994 International Conference on Very Large Data Bases, Santiago, Chile, 487-499.

Renxu Sun, Chai-Huat Ong, Tat-Seng Chua. 2006. *Mining dependency relations for query expansion in passage retrieva**l.*** Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Pages: 382 – 389.

Richard Sproat and Chilin Shih. 2002. *Corpus-Based Methods in Chinese Morphology and Phonology*.

Robertson, S.E. and K. Sparck Jones. 1976. *Relevance Weighting of Search Terms*. Journal of the American Society for Information Science, 27(3): 129-146.

Rocchio, J. 1997. *Relevance feedback in information retrieval. In the Smart Retrieval System: Experiment in Automatic Document Processing*, Pages 313-323.

Smeaton, A. F. and Van Rijsbergen, C. J. 1983. *The retrieval effects of query expansion on a feedback document retrieval system*.Computer Journal, 26(3): 239-246.

Wei-Yun Ma, Keh-Jiann Chen. 2002. *Introduction to CKIP Chinese Word Segmentation System for the First International Chinese Word Segmentation Bakeoff.* Institute of Information science, Academia Sinica.

Xinjing Wang, Wen Liu and Yong Qin. 2006. *A Search-based Chinese Word Segmentation Method*. Proceedings of the 16th international conference on World Wide Web.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 79-87

Automatic Event Reference Identification

Olivia March
CSSE
University of Melbourne
Victoria 3010
Australia
oliviacm@csse.unimelb.edu.au

Timothy Baldwin
NICTA Victoria Research Laboratories
University of Melbourne
Victoria 3010
Australia
tim@csse.unimelb.edu.au

Abstract

Event reference identification is often treated as a sentence level classification task. However, several different event references can occur within a single sentence. We present a set of experiments involving real world event reference identification at the word level in newspaper and newswire documents, addressing the issue of effective text representation for classification of events using support vector machines. Our final system achieved an F-score of 0.764, significantly exceeding that of our baseline system. Additionally we achieved a marginally higher performance than a more complex comparable system.

1 Introduction

Multi-document summarization seeks to combine the salient information from multiple independent documents into a single coherent summary. The Document Understanding Conference (DUC) shared tasks are a measured attempt to apply multi-document summarization to newswire document sets, to produce a fluent 250 word summary (Dang, 2006). The resultant summaries are then scored and assessed, depending on the specifics of the task.

Newswire and newspaper document sets such as those used in DUC often describe the same series of events across multiple sources, for example, media coverage of an election in Bolivia. Here, a time-line can be a more effective way of structuring a multi-document summary than a traditional paragraph-style summary. More generally, historical and biographical information is frequently presented as a time-line of events, and the ability to produce such time-lines from independent sources is valuable for online content providers such as Wikipedia. The following is an illustration of the utility of time-lines in analysing topics (events highlighted in **bold** to illustrate their importance in time-line generation):

> The **Russian Revolution** in 1917 was **triggered** by a combination of **economic breakdown**, war weariness, and discontent with the autocratic system of government, and it first **brought** a coalition of liberals and moderate socialists to **power**, but their failed policies led to **seizure** of power by the Communist Bolsheviks on October 25.

Existing summarization systems, such as the DUC 07 main task system of Stokes et al. (2007), are effective at general summary tasks but have difficulty with event based summaries. For example, Stokes et al. (2007) performs well on general questions centred on a specified topic such as:

> What attacks have occurred against tourists in Luxor, Egypt?

but not so well on topics requiring a listing of events, such as:

> What have been the most recent significant events in the life and career of actress Angelina Jolie?

In this paper, we identify the most effective representation of text for real-world event reference identification in newspaper and newswire text using support vector machines (SVMs). The work presented here is the basis of an event-based summary system which will in turn form a component of the Stokes et al. (2007) system, where each question will be classified as an event-based question or a general question. The corresponding summary will then be generated by the appropriate sub-system, i.e. the original general-purpose system or a dedicated event-based system.

In creating an event-based time-line, three main tasks need to be considered: (1) how to identify and group references to events in text; (2) how to ground the identified events to a timestamp; and finally (3) how to identify salient events for inclusion in the final fixed-length time-line summary. In this paper we specifically focus on the identification of event references in text.

Often the task of event identification is linked to that of temporal resolution, whereby a system resolves the time or date of the occurrence of an event. Many systems treat events as any sentence containing an event reference and focus efforts on identifying temporal cues in the text. As there can be multiple event references within a sentence we identify events at the word level.

The following example is an illustration of the type of event-based identification we aim to achieve in this paper, relative to a plain-text input sentence (events highlighted in **bold**):

> Amid **reports** that thousands of Iraqi soldiers had **surrendered**, administration aides were also upbeat in private, with one even **talking** of **victory** within a week.

The focus of this paper is an exploration of the most appropriate feature representation to use in a purely statistical approach to word-level event identification in newswire text.

In the remainder of this paper, we review the notion of "event" in the literature, and present an overview of the different approaches to event identification. We then describe our experimental methodology, including the TimeBank corpus. Finally we describe a comparative evaluation.

2 Related Work

2.1 Event Definition

The definition of an event varies in granularity depending on the desired application of event extraction.

The Scenario Template task of the Message Understanding Conference (MUC) (Grishman and Sundheim, 1996) relied on specified domain-dependent templates which define events for the purpose of information extraction. The events that are extracted depend on the templates and documents supplied. MUC consists of domain-specific scenario (topic) based templates that systems are required to fill. An example domain is financial news in the Wall Street Journal, e.g. with the scenario of change in the corporate executive management personnel of an organization (Grishman and Sundheim, 1996).

In the context of Topic Detection and Tracking (TDT) (Fiscus et al., 1998), an event corresponds to an instance of a topic identified at the document level, where the aim is to cluster documents on the same topic rather than individual events within the document. For instance, given an earthquake in the Solomon Islands on August 13, in TDT *earthquakes* would be the topic and this particular earthquake instance would be the event. The task would then be to cluster all documents that refer to this specific earthquake.

The aim of the Automatic Content Extraction (ACE) (Doddington et al., 2004) task is to identify and classify events of predefined semantic types. An event reference may be a noun, verb or phrase, however events are tagged at the sentence level (LDC, 2005). In the above-mentioned documents referring to the Solomon Islands earthquake, the ACE data would have each reference marked as an event, in addition to announcements made by officials and any other words/phrases referring to events within the documents.

Setzer (2001) (and later TimeML (Pustejovsky et al., 2005)) define an event as something that happens with a defined beginning and ending within the scope of a document. By comparison, a state is something that holds between entities without a definite beginning or end. The identification of events is based on verb and noun analysis similar in granularity to that in the ACE task. ACE and TimeML differ primarily on the semantic class labels for the events, and the annotated attributes and entities associated with each event. Where TimeML has seven semantic class labels (Occurrence, State, Reporting, I-Action, I-State, Aspectual and Perception), ACE has five more specific labels (Destruction/Damage, Movement, Creation/Improvement, Transfer of Possession or

Control and Interaction of agents) (LDC, 2005).

For our purposes in this research, we define an event to be something that happens or occurs within the context of the document, similar to Setzer (2001). However, we do not limit events to being denoted by only verbs or nouns. We assume that an event reference is contained within a sentence and does not cross sentence boundaries. Section 3.1 contains a detailed description of how an event is defined in our development data.

2.2 Previous Work in Event Classification

Approaches to event extraction vary from ontology- and frame-based systems to combined rule-based and statistical methods. In this section we describe a number of representative approaches to event extraction from the literature.

The REES system (Aone and Ramos-Santacruz, 2000) uses hand-crafted event and relation ontologies to extract 100 relation and event types, 61 of which are events. It is purported to be extensible, however only so far as the ontology is extended. Events and relations relate to templates with type, person, time, place, etc, as defined in the ontology. The system consists of three parts: a tagging module (containing NameTagger, NPTagger and EventTagger), a rule-based co-reference resolution module, and a template generation module (described as a non-hard-coded approach that uses declarative rules to generate and merge templates automatically to "achieve portability"). MUC style data was used for training and testing. The system achieved a 0.70 F-score over 26 event types.

Jones (2003) applied Hidden Markov Models to the automatic generation of scripts to identify events. HMMs were used to capture correlations of significant events in text: the nodes correspond to events in the text, and the arcs indicate which clauses occur together in text. The context of neighbouring events and textual similarity are used to decide whether or not to group clauses.

As a component of TARSQI, aimed at interpreting entities and events in temporally-based questions, Verhagen and Mani (2005) developed the EVITA event recognition tool. EVITA is used for event recognition in newswire text and analysis of grammatical features, such as tense and aspect. Events are identified using lexical analysis, context analysis of verbs, lexical lookup of adjectival events, and machine learning to determine whether an ambiguous noun is used in an event sense. They combine linguistic and statistical methods to obtain a 74.0% precision and 87.3% recall (Sauir et al., 2005). Time references and events are defined by the TimeML annotation scheme.

Preprocessing of the data was carried out using the Alembic Workbench for part of speech tagging, lematizing and chunking. A shallow parser is used to retrieve event referring expressions which are conveyed by verbs, nouns and adjectives. Verb-based events are identified by lexical lookup and contextual parsing of the verbal chunks. Events denoted by nouns are identified via WordNet and disambiguation with a Bayesian classifier trained on SemCor. Adjectives are tagged as events when they come at the head of a predictive complement, such as:

> A Philippine volcano, **dormant** for six centuries, ... (TimeML, 2006)

EVITA also classifies events semantically, which we do not consider in this paper.

Bethard and Martin (2006) describe a system to identify events and their semantic class for the purposes of question answering. They define events as a description of situations that involve some internal structure, as opposed to states which describe a situation that is static or unchanging. Bethard and Martin (2006) view event identification as a classification task. Their system STEP is able to identify events with a precision of 82% and recall of 71%. They use a suite of syntactic and semantic features as input to YamCha — a general purpose chunker — and TinySVM. Each word in the document is classified as either beginning (B), inside (I) or outside (O) an event.

3 Experimental Methodology

3.1 Data: TimeML and TimeBank

Given that our notion of event closely mirrors that of TimeBank, we were able to use the TimeBank 1.1 corpus for system development and evaluation. TimeBank 1.1 consists of 186 news articles marked up with TimeML standard 1.1. The articles are sourced primarily from the Wall Street

Journal and Associated Press newswire articles. They contain EVENT tags denoting events at the phrase and word level.

TimeML is a specification language for event and temporal expressions in text (Pustejovsky et al., 2005), and builds on TIDES TIMEX2 and the temporal annotation language presented in Setzer (2001).

TimeML addresses four basic problems in event-time identification:

- Time stamping of events (anchor event in time).
- Chronological ordering of events with respect to one another.
- Reasoning with contextually under-specified temporal expressions (e.g. *last week*).
- Reasoning about the persistence of events (how long does an event or outcome of an event last).

TimeML has four major event-time structures: EVENT (see Section 3.1.1), TIMEX3 (used to denote explicit time references), SIGNAL (used primarily to annotate function words indicating how temporal objects relate to one another) and LINK (denoting the relationship between an event and other events or temporal expressions).

3.1.1 EVENT

Events are a cover term for situations that happen or occur, and can be punctual or last for a time period. Events are generally expressed by tensed or untensed verbs, nominalisations, adjectives, predicative clauses or prepositional phrases. An EVENT in TimeML contains several attributes: a unique identifier, a class, tense, aspect and optionally another event. It may also contain another event or time to which it is temporally related.

There are seven event class types (each of which is illustrated with a set of examples):

- *Occurrence:* die, crash, build, merger, sell
- *State:* on board, kidnaped, in love
- *Reporting:* say, report, announce
- *I-Action:* attempt, try, promise, offer
- *I-State:* believe, intend, want
- *Aspectual:* begin, finish, stop, continue
- *Perception:* see, hear, watch, feel

We currently disregard the event classes and use a binary classification, identifying each word as an event or non-event.

3.2 System Description

We preprocessed the TimeBank data to remove extraneous XML tags, decompose it into sentences and words, and discard punctuation. Finally we assign each word with a POS tag using the Stanford POS tagger (Toutanova et al., 2003).

In our experiments, we use three classification algorithms: decision trees, naive Bayes and support vector machines. The chosen training and evaluation platform was WEKA (Witten and Frank, 2005). Stratified 10-fold cross-validation was used throughout, where the labeled text fragments were randomly allocated to different combinations of training and testing splits. We report on the average performance over the 10 runs.

Preliminary experiments over the three classification algorithms indicated that SVMs were more effective in the classification of event references in text. BSVM (Hsu and Lin, 2002) was used for subsequent experiments, again using 10-fold cross-validation.

In Section 3.3, we present a range of different feature types, designed to explore the impact of different information sources on event reference classification.

In our classifiers, we look first at sentence-level event classification (i.e. does a given sentence contain one or more event references) and then word-level event classification (i.e. is a given word an event reference). The sentence-level classifier is used to filter out the non-event sentences, and only event-containing sentences are passed on to the word-level classifier.

3.3 Features

We use a basic word feature vector to represent the words being classified. Each distinct word in the input text corresponds to a unique feature. The text is transformed into a vector of N values where N is the number of distinct words in the TimeBank corpus. The following are common text representation techniques in text classification, the effects of which we explore in the context of event identification.

- **Context Window**
  For identification of events at the word level we explore the use of a context window of the zero to three preceding words, in addition to the word being classified.
- **Feature Representation**
  We also look at the effect of representing the current and context window words as a bag of word types (binary representation), or list of position-specified words (representing each word by its positional value [current word = 1, previous word = 2, etc; other words = 0]).
- **Stop Words**
  Stop words are short, frequently occurring words such as *and*, *or*, *in*, *of*. Stop word removal reduces the feature space with generally little impact on classification performance in document categorisation tasks, but has been shown to be detrimental in sentence-level classification tasks (Khoo et al., 2006). In our research, we use the stop word list of van Rijsbergen (1979).
- **POS**
  Parts of speech (POSs) are assigned to each word using the Stanford maxent POS tagger, and optionally used as an alternative to word features for context words (i.e. each context word is represented as its POS).
- **Feature Generalisation**
  We optionally group numbers, times and named entities into single features. Numbers and dollar amounts were identified using regular expressions, and named entities were tagged using the Alembic Workbench (Aberdeen et al., 1995).

3.4 Evaluation Method

We use F-score as our primary evaluation metric, as defined below relative to Precision and Recall:

$$\text{Precision} = \frac{\textit{\# words correctly classified as events}}{\textit{\# words classified as events}} \quad (1)$$

Model	Precision	Recall	F-Score
Baseline	0.800	0.550	0.652
STEP	0.820	0.706	0.759
EVITA	0.740	0.873	0.801

Table 1: Performance of comparable event identification systems and our baseline system, as evaluated over the TimeBank data

$$\text{Recall} = \frac{\textit{\# words correctly classified as events}}{\textit{\# words that are events}} \quad (2)$$

$$\text{F-Score} = \frac{2 \times \textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (3)$$

Our baseline takes the form of a unigram tagger, where each word is labeled with its most frequent label. Table 1 shows the performance of the baseline system, in addition to the the comparable STEP (Aone and Ramos-Santacruz, 2000) and EVITA (Sauir et al., 2005) results — as described in Section 2.2 — both of which use the TimeBank data in the development and testing of their systems.

4 Results

4.1 Sentence Level Classification of Events

In this section we first present the results of our sentence-level event identification system, where each sentence is classified as containing one or more event references, or alternatively containing no event references. Three learners — C4.5 decision tree, SVM and naive Bayes — were run over a random subset of the TimeBank data, the results of which are shown in Table 2.

All three methods performed exceptionally well, at an F-score of or close to 1. This level of performance is due to 82% of the sentences in the corpus containing an event reference, and most of these containing more than one event reference. Therefore only one of the many event references within a sentence needs to be identified for the sentence to be correctly classified.

The naive Bayes classifier and decision tree models proved difficult to run over the entire TimeBank corpus due to resource constraints. As a result, we chose to use BSVM (Hsu and Lin, 2002) for all subsequent sentence-level experiments.

Algorithm	Precision	Recall	F-Score
Naive Bayes	0.995	0.998	0.996
SVM	1.000	1.000	1.000
C4.5	1.000	1.000	1.000

Table 2: Performance of sentence level classification with stop word removal and feature generalisation

As there can be multiple references to different events within a single sentence, sentence level classification is not fine-grained enough for event clustering and extraction in the context of event-based summary generation. The sentence classifier is instead used as a pre-filter for word-level event classification, as discussed in the following section.

4.2 Word-level Classification of Events

As stated above, in classifying individual words for event reference, we first use the sentence-level classifier to remove sentences that do not contain events. This proved to have no effect on the accuracy of the word level system, but did offer a slight reduction in the time required by the word-level classifier.

Both the naive Bayes classifier and C4.5 decision tree resulted in majority class classifiers (see Table 1), where all words were tagged as non-events. The SVM, on the other hand, produced a variety of results, and thus forms the basis of all results presented in this section.

Below, we will discuss our findings into the effects of stop word removal, feature representation, context window size, POS tagging and feature generalisation on word-level event classification.

4.2.1 Context Window Size

Increasing the context window, using a binary feature representation, decreased the accuracy of the classifier, as shown in the leftmost segment of Figure 1. The zero word context improves only 1% with an increase from 15 to 18 thousand training instances compared with an improvement of 7% for the three word prior context model. At this point, we have been unable to improve over the baseline system.

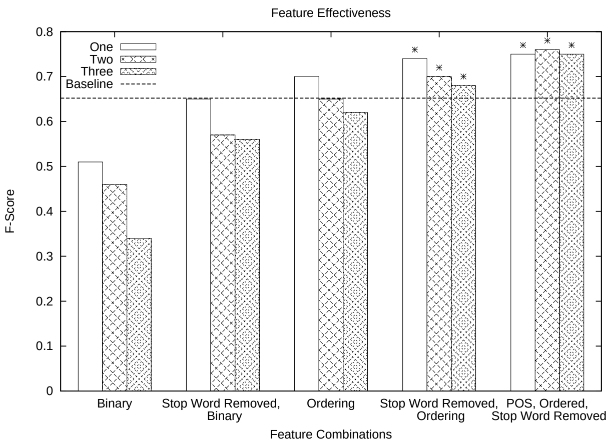


Figure 1: The F-score of models trained over different feature combinations. (* = statistically significant difference: two-tailed paired t-test, $p < 0.05$)

4.2.2 Feature Representation

Replacing the binary bag of words representation with a position-specified binary word representation has a dramatic impact on our results, as indicated in the middle set of results in Figure 1. Maintaining the ordering of 2 word prior context has an F-score of 0.68, compared to a binary representation of 2 word prior context with an F-score of 0.41.

4.2.3 Stop Words

Khoo et al. (2006) found that stop word removal had a detrimental effect on classification at the sentence level. Our results, however, demonstrate that stop word removal offers a slight increase in the accuracy of the word-level classifiers. Table 3 shows the results obtained using SVM to identify event references at the word level using different combinations of text representation techniques. As we can see from the table, stop word removal had a positive effect on the system irrespective of whether a binary or word order representation was used. With a one word POS context window and feature generalisation, stop word removal increases the F-score by 0.043. Similarly, using a binary representation and grouping, stop word removal increases the F-score by 0.069.

4.2.4 Part Of Speech

Including the POS tags of the context and word being classified increases performance of

Feature combination	Precision	Recall	F-score
Word order and grouping	0.712	0.675	0.693
Word order, grouping and stop word removal	0.761	0.712	0.736
Binary and grouping	0.623	0.462	0.531
Binary, grouping and stop word removal	0.673	0.542	0.600

Table 3: The results of word-level event identification using an SVM and different combinations of features

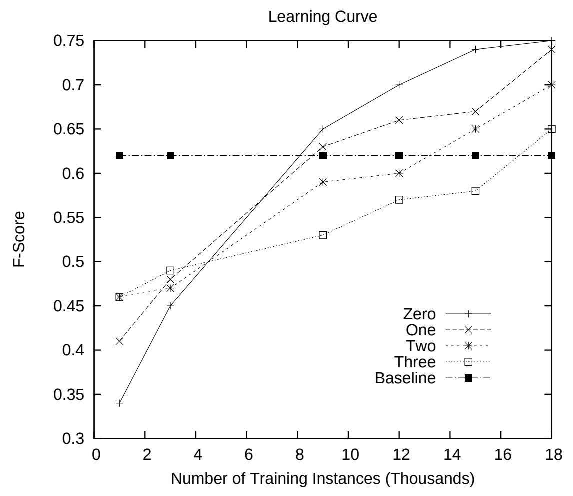


Figure 2: The learning curves for the models of different context window sizes, with feature generalisation and stop word removal

the overall system. However, the greatest increase comes from the inclusion of the preceding context represented as a part of speech, without context words. This facilitates the generalisation of the patterns leading up to the event reference. Note that the word being tagged is retained as a binary feature, as its exclusion would leave a POS pattern feature that is too general given the limited combination of less than fifty distinct POS tags.

4.2.5 Feature Generalisation

Grouping specific types of words such as numbers, named entities and time references into a single feature proved an effective way of reducing the feature space. However, closer analysis of the word-level event classification results showed that some of the terms being grouped together occurred as events whereas others did not. There are 610 NUMBER references in the TimeBank data. 247 of these are an event or part thereof and 363 are not, leading to most NUMBER references being tagged as non-events. For example, monetary amounts can be a state (type of event) if they are a milestone for a company, or alternatively they can simply be an amount lost by the company.

> The insurer's earnings from commercial property lines **fell** 59% in the latest quarter, while it **lost** $7.2 million in its personal property business, compared with earnings of **$6.1 million**[1]a year ago.

The other groupings used, TIME and NAMED ENTITY, are never event references in the TimeBank corpus, and thus their grouping reduces the feature space without loss of the ability to disambiguate. Further investigation into a more fine-grained grouping of numerical terms is required before grouping numbers is included in our final system.

4.3 Discussion

Sentence-level classification proved a simple task with an almost perfect classification accuracy. The use of a sentence-level classifier for prior filtering of the word-level event system had no effect of the overall F-score.

Our final system combined POS representation of the two word prior context with context positional information and stop word removal to achieve an F-score of 0.764, exceeding our baseline of 0.653 and marginally exceeding the performance of the comparable STEP system with an F-score of 0.759. Over 50% of incorrectly classified words resulted from event words which occur only once in the TimeBank corpus. A further 20% are words that appear as both events and non-events in the training data. 7% of the remaining errors result from incorrect tokenisation dur-

[1]States that describe circumstances in which something obtains and holds true are called predictive states. The validity of this is dependent on the document creation time, and includes some quantitative statements such as those that appear in financial journals (TimeML, 2006)[pg14].

ing preprocessing. There are 1540 words in the TimeBank data that occur only once as events.

To investigate the impact of the amount of data on our results, we generate learning curves for word order-sensitive feature representation with stop word removal and feature generalisation, over diminishing amounts of the training data used in cross-validation. The curves are presented in Figure 2 over different context word window sizes. As we can see, the models based on larger context windows are still climbing steeply over the final increment of training instances, while the zero-word context model appears to be flattening out. As such, we would expect that an increase in the number of training instances to improve the performance of the models with larger context windows, although it remains to be seen how much they would improve relative to the smaller word windows.

Maintaining word order has the biggest effect on the performance of the classification system. Representing the preceding context as POS rather than words also increases the accuracy of the overall system and, contrary to expectations, the removal of stop words marginally increased the performance. As indicated by the learning curve (Figure 2), increasing the number of training instances has the potential to improve the performance of the models which include context.

The two-word POS context model misclassifies a different subset of words to the three-word POS context model. Therefore, combining different models may increase the performance of the overall system.

The majority of errors came from previously unseen words in the training data. Future work will thus focus on how the system can better handle unseen words.

5 Future Work

We intend to look at the effect verb types obtained from WordNet (Fellbaum, 1998) or VERBOCEAN (Chklovski and Pantel, 2004) have on word-level event identification, to assist in better handling of unseen words. We will also look at how the system performance scales over a larger data set.

6 Conclusion

We presented a set of experiments involving real-world event reference identification at the word level in newspaper and newswire documents. We addressed the issue of effective text representation for classification of events using support vector machines. Our final system combined two word POS prior context, with positional information and stop word removal. It achieved an F-score of 0.764, significantly exceeding that of our baseline system with an F-score of 0.653.

Acknowledgments

NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme.

References

John Aberdeen, John D. Burger, David S. Day, Lynette Hirschman, Robinson, Patricia, and Marc B. Vilain. 1995. MITRE: Description of the Alembic system used for MUC-6. In *Proceedings of MUC-6*.

Chinatsu Aone and Mila Ramos-Santacruz. 2000. REES: A large-scale relation and event extraction system. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, USA.

Steven Bethard and James Martin. 2006. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Emprical Methods in Natural Language Processing*, Sydney, Australia.

Timothy Chklovski and Patrick Pantel. 2004. VERBOCEAN: Mining the web for fine-grained semantic verb relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain.

Hoa Trang Dang. 2006. Overview of DUC 2006. In *Proceedings of the Document Understanding Conference Workshop*, New York, USA.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. Automatic content extraction (ACE) program - task definitions and performance measures. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*. Lisbon, Portugal.

Christiane Fellbaum. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, first edition.

Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. 1998. NIST's 1998 topic detection and tracking evaluation. In *Proceedings of the DARPA Broadcast News Workshop*.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference 6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics*, Copenhagen, Denmark.

Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison on methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425.

Dominic R. Jones. 2003. Identifying events using similarity and context. In *Proceedings of the Seventh Conference on Natural Language Learning*, Edmonton, Canada.

Anthony Khoo, Yuval Marom, and David Albrecht. 2006. Experiments with sentence classification. In *Proceedings of the 2006 Australasian Language Technology Workshop*, Sydney, Australia.

LDC. 2005. ACE (automatic content extraction) English annotation guidelines for events.

James Pustejovsky, Robert Ingria, Roser Sauri, Jose Castano, Jessica Littman, Rob Gaizauskas, Andrea Setzer, Graham Katz, and Inderjeet Mani. 2005. The specification language TimeML. In *The Language of Time: A Reader*, chapter 27. Oxford University Press.

Roser Sauir, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. EVITA: A robust event recognizer for QA systems. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 700–707, Vancouver, Canada.

Andrea Setzer. 2001. *Temporal Information In Newswire Articles: An Annotation Scheme and Corpus Study*. Ph.D. thesis, University of Sheffield.

Nicola Stokes, Jiawen Rong, and Lawrence Cavedon. 2007. NICTA's update and question-based summarisation systems at DUC 2007. In *Proceedings of the Document Understanding Conference Workshop*, Rochester, USA.

TimeML. 2006. TimeML annotation guidelines version 1.2.1.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259, Edmonton, Canada.

C.J. Keith van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann, 2nd edition.

Marc Verhagen and Inderjeet Mani. 2005. Automating temporal annotaion with TARSQI. In *Proceedings of the ACL 2005 Interactive Poster and Demonstration Sessions*, Ann Arbor, USA.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Fransisco, second edition.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 88-96

Comparing the value of Latent Semantic Analysis on two English-to-Indonesian lexical mapping tasks

Eliza Margaretha
Faculty of Computer Science
University of Indonesia
Depok, Indonesia
elm40@ui.edu

Ruli Manurung
Faculty of Computer Science
University of Indonesia
Depok, Indonesia
maruli@cs.ui.ac.id

Abstract

This paper describes an experiment that attempts to automatically map English words and concepts, derived from the Princeton WordNet, to their Indonesian analogues appearing in a widely-used Indonesian dictionary, using Latent Semantic Analysis (LSA). A bilingual semantic model is derived from an English-Indonesian parallel corpus. Given a particular word or concept, the semantic model is then used to identify its neighbours in a high-dimensional semantic space. Results from various experiments indicate that for bilingual word mapping, LSA is consistently outperformed by the basic vector space model, i.e. where the full-rank word-document matrix is applied. We speculate that this is due to the fact that the 'smoothing' effect LSA has on the word-document matrix, whilst very useful for revealing implicit semantic patterns, blurs the cooccurrence information that is necessary for establishing word translations.

1 Overview

An ongoing project at the Information Retrieval Lab, Faculty of Computer Science, University of Indonesia, concerns the development of an Indonesian WordNet[1]. To that end, one major task concerns the mapping of two monolingual dictionaries at two different levels: bilingual word mapping, which seeks to find translations of a lexical entry from one language to another, and bilingual concept mapping, which defines equivalence classes over concepts defined in two language resources. In other words, we try to automatically construct two variants of a bilingual dictionary between two languages, i.e. one with sense disambiguated entries and one without.

In this paper we present an extension to LSA into a bilingual context that is similar to (Rehder et al., 1997; Clodfelder, 2003; Deng and Gao, 2007), and then apply it to the two mapping tasks described above, specifically towards the lexical resources of Princeton WordNet (Fellbaum, 1998), an English semantic lexicon, and the *Kamus Besar Bahasa Indonesia* (KBBI)[2], considered by many to be the official dictionary of the Indonesian language.

We first provide formal definitions of our two tasks of bilingual word and concept mapping (Section 2) before discussing how these tasks can be automated using LSA (Section 3). We then present our experiment design and results (Sections 4 and 5) followed by an analysis and discussion of the results in Section 6.

2 Task Definitions

As mentioned above, our work concerns the mapping of two monolingual dictionaries. In our work, we refer to these resources as WordNets due to the fact that we view them as semantic lexicons that index entries based on meaning. However, we do not consider other semantic relations typically associated with a WordNet such as hypernymy, hyponymy, etc. For our purposes, a WordNet can be

[1] http://bahasa.cs.ui.ac.id/iwn

[2] The KBBI is the copyright of the Language Centre, Indonesian Ministry of National Education.

formally defined as a 4-tuple (C, W, χ, ω) as follows:

- A concept $c \in C$ is a semantic entity, which represents a distinct, specific meaning. Each concept is associated with a gloss, which is a textual description of its meaning. For example, we could define two concepts, c_1 and c_2, where the former is associated with the gloss "*a financial institution that accepts deposits and channels the money into lending activities*" and the latter with "*sloping land (especially the slope beside a body of water*".
- A word $w \in W$ is an orthographic entity, which represents a word in a particular language (in the case of Princeton WordNet, English). For example, we could define two words, w_1 and w_2, where the former represents the orthographic string *bank* and the latter represents *spoon*.
- A word may convey several different concepts. The function $\chi: W \to P(C)$ returns all concepts conveyed by a particular word. Thus, $\chi(w)$, where $w \in W$, returns $C_w \subset C$, the set of all concepts that can be conveyed by w. Using the examples above, $\chi(w_1) = \{c_1, c_2\}$.
- Conversely, a concept may be conveyed by several words. The function $\omega: C \to P(w)$ returns all words that can convey a particular concept. Thus, $\omega(c)$, where $c \in C$, returns $W_c \subset W$, the set of all words that convey c. Using the examples above, $\omega(c_1) = \omega(c_2) = \{w_1\}$.

We can define different WordNets for different languages, e.g. $N^e = (C^e, T^e, \chi^e, \omega^e)$ and $N^i = \left(C^i, T^i, \chi^i, \omega^i\right)$. We also introduce the notation w_i^x to denote word i in W^x and c_j^x to denote concept j in C^x. For the sake of our discussion, we will assume N^e to be an English WordNet, and N^i to be an Indonesian WordNet.

If we make the assumption that concepts are language independent, N^e and N^i should theoretically share the same set of universal concepts, C. In practice, however, we may have two WordNets with different conceptual representations, hence the distinction between C^e and C^i. We introduce the relation $E : C^e \times C^i$ to denote the explicit mapping of equivalent concepts in C^e and C^i.

We now describe two tasks that can be performed between N^e and N^i, namely bilingual concept mapping and bilingual word mapping.

The task of bilingual concept mapping is essentially the establishment of the concept equivalence relation E. For example, given the example concepts in Table 1, bilingual concept mapping seeks to establish $E = \{(c_1^e, c_1^i), (c_1^e, c_2^i), (c_2^e, c_3^i), (c_3^e, c_4^e)\}$.

Concept	Word	Gloss	Example
c_1^e	w_{time}^e	an instance or single occasion for some event	*"this **time** he succeeded"*
c_2^e	w_{time}^e	a suitable moment	*"it is **time** to go"*
c_3^e	w_{time}^e	a reading of a point in time as given by a clock (a word signifying the frequency of an event)	*"do you know what **time** it is?"*
c_1^i	w_{kali}^i	kata untuk menyatakan kekerapan tindakan (a word signifying a particular instance of an ongoing series of events)	*"dalam satu minggu ini, dia sudah empat **kali** datang ke rumahku" (this past week, she has come to my house four **times**)*
c_2^i	w_{kali}^i	kata untuk menyatakan salah satu waktu terjadinya peristiwa yg merupakan bagian dari rangkaian peristiwa yg pernah dan masih akan terus terjadi (a word signifying a particular instance of an ongoing series of events)	*"untuk **kali** ini ia kena batunya" (this **time** he suffered for his actions)*
c_3^i	w_{waktu}^i	saat yg tertentu untuk melakukan sesuatu (a specific time to be doing something)	*"**waktu** makan" (eating **time**)*
c_4^i	w_{jam}^i	saat tertentu, pada arloji jarumnya yg pendek menunjuk angka tertentu dan jarum panjang menunjuk angka 12 (the point in time when the short hand of a clock points to a certain hour and the long hand points to 12)	*"ia bangun **jam** lima pagi" (she woke up at five **o'clock**)*
c_5^i	w_{kali}^i	sebuah sungai yang kecil (a small river)	*"air di **kali** itu sangat keruh" (the water in that **small river** is very murky)*

Table 1. Sample Concepts in C^e and C^i

The task of bilingual word mapping is to find, given word $w_x^e \in W^e$, the set of all its plausible translations in W^i, regardless of the concepts being conveyed. We can also view this task as computing the union of the set of all words in W^i that convey the set of all concepts conveyed by w_x^e. Formally, we compute the set $\{w_y^i : w_y^i \in \omega^i(c^i)$ where $(c^e, c^i) \in E$ and $c^e \in \chi^e(w_x^e)\}$.

For example, in Princeton WordNet, given w_{time}^e (i.e. the English orthographic form time), $\chi^e(w_{time}^e)$ returns more than 15 different concepts, among others $\{c_1^e, c_2^e, c_3^e\}$ (see Table 1).

In Indonesian, assuming the relation E as defined above, the set of words that convey c_1^i, i.e. $\omega^i(c_1^i)$, includes w_{kali}^i (as in "kali ini dia berhasil" = "this time she succeeded").

On the other hand, $\omega^i(c_3^i)$ may include w_{waktu}^i (as in "ini waktunya untuk pergi" = "it is time to go") and w_{saat}^i (as in "sekarang saatnya menjual saham" = "now is the time to sell shares"), and lastly, $\omega^i(c_4^i)$ may include w_{jam}^i (as in "apa anda tahu jam berapa sekarang?" = "do you know what time it is now?").

Thus, the bilingual word mapping task seeks to compute, for the English word w_{time}^e, the set of Indonesian words $\{w_{kali}^i, w_{waktu}^i, w_{saat}^i, w_{jam}^i, \dots\}$. Note that each of these Indonesian words may convey different concepts, e.g. $\chi^i(w_{kali}^i)$ may include c_5^i in Table 1.

3 Automatic mapping using Latent Semantic Analysis

Latent semantic analysis, or simply LSA, is a method to discern underlying semantic information from a given corpus of text, and to subsequently represent the contextual meaning of the words in the corpus as a vector in a high-dimensional semantic space (Landauer et al., 1998). As such, LSA is a powerful method for word sense disambiguation.

The mathematical foundation of LSA is provided by the Singular Value Decomposition, or SVD. Initially, a corpus is represented as an $n \times m$ word-passage matrix M, where cell $[n, m]$ represents the occurrence of the n-th word in the m-th passage. Thus, each row of M represents a word and each column represents a passage. The SVD is then applied to M, decomposing it such that $M = USV^T$, where U is an $m \times m$ matrix of left singular vectors, V^T is an $n \times n$ matrix of right singular vectors, and Σ is an $n \times m$ matrix containing the singular values of M.

Crucially, this decomposition factors M using an orthonormal basis that produces an optimal reduced rank approximation matrix (Kalman, 1996). By reducing dimensions of the matrix irrelevant information and noise are removed. The optimal rank reduction yields useful induction of implicit relations. However, finding the optimal level of rank reduction is an empirical issue.

LSA can be applied to exploit a parallel corpus to automatically perform bilingual word and concept mapping. We define a parallel corpus P as a set of pairs $p = (d_e, d_i)$, where d_e is a document written in the language of N^e, and d_i is its translation in the language of N^i.

Intuitively, we would expect that if two words w_x^e and w_x^i consistently occur in documents that are translations of each other, but not in other documents, that they would at the very least be semantically related, and possibly even be translations of each other. For instance, imagine a parallel corpus consisting of news articles written in English and Indonesian: in English articles where the word *Japan* occurs, we would expect the word *Jepang* to occur in the corresponding Indonesian articles.

This intuition can be represented in a word-document matrix as follows: let M_E be a word-document matrix of m English documents and n_E English words, and M_I be a word-document matrix of m Indonesian documents and n_I Indonesian words. The documents are arranged such that, for $1 \le j \le m$, the English document represented by column j of M_E and the Indonesian document represented by column j of M_I form a pair of translations. Since they are translations, we can view them as occupying exactly the same point in semantic space, and could just as easily view column j of both matrices as representing the union, or concatenation, of the two articles.

Consequently, we can construct the bilingual word-document matrix

$$M = \begin{bmatrix} M_E \\ M_I \end{bmatrix}$$

which is an $(n_E + n_I) \times m$ matrix where cell $[i, j]$ contains the number of occurrences of word i in article j. Row i forms the semantic vector of, for $i \le n_E$, an English word, and for $i > n_E$, an Indonesian word. Conversely, column j forms a vector representing the English and Indonesian words appearing in translations of document j.

This approach is similar to that of (Rehder et al., 1997; Clodfelder, 2003; Deng and Gao, 2007). The SVD process is the same, while the usage is different. For example, (Rehder et al., 1997) employ SVD for cross language information retrieval. On the other hand, we use it to accomplish word and concept mappings.

LSA can be applied to this bilingual word-document matrix. Computing the SVD of this matrix and reducing the rank should unearth implicit patterns of semantic concepts. The vectors representing English and Indonesian words that are closely related should have high similarity; word translations more so.

To approximate the bilingual word mapping task, we compare the similarity between the semantic vectors representing words in W^e and W^i. Specifically, for the first n_E rows in M which represent words in W^e, we compute their similarity to each of the last n_I rows which represent words in W^i. Given a large enough corpus, we would expect all words in W^e and W^i to be represented by rows in M.

To approximate the bilingual concept mapping task, we compare the similarity between the semantic vectors representing concepts in C^e and C^i. These vectors can be approximated by first constructing a set of textual context representing a concept c. For example, we can include the words in $\omega(c)$ together with the words from its gloss and example sentences. The semantic vector of a concept is then a weighted average of the semantic vectors of the words contained within this context set, i.e. rows in M. Again, given a large enough corpus, we would expect enough of these context words to be represented by rows in M to form an adequate semantic vector for the concept c.

4 Experiments

4.1 Existing Resources

For the English lexicon, we used the most current version of WordNet (Fellbaum, 1998), version 3.0[3]. For each of the 117659 distinct synsets, we only use the following data: the set of words belonging to the synset, the gloss, and example sentences, if any. The union of these resources yields a set 169583 unique words.

For the Indonesian lexicon, we used an electronic version of the KBBI developed at the University of Indonesia. For each of the 85521 distinct word sense definitions, we use the following data: the list of sublemmas, i.e. inflected forms, along with gloss and example sentences, if any. The union of these resources yields a set of 87171 unique words.

Our main parallel corpus consists of 3273 English and Indonesian article pairs taken from the ANTARA news agency. This collection was developed by Mirna Adriani and Monica Lestari Paramita at the Information Retrieval Lab, University of Indonesia[4].

A bilingual English-Indonesia dictionary was constructed using various online resources, including a handcrafted dictionary by Hantarto Widjaja[5], kamus.net, and Transtool v6.1, a commercial translation system. In total, this dictionary maps 37678 unique English words to 60564 unique Indonesian words.

4.2 Bilingual Word Mapping

Our experiment with bilingual word mapping was set up as follows: firstly, we define a collection of article pairs derived from the ANTARA collection, and from it we set up a bilingual word-document matrix (see Section 3). The LSA process is subsequently applied on this matrix, i.e. we first compute the SVD of this matrix, and then use it to compute the optimal k-rank approximation. Finally, based on this approximation, for a randomly chosen set of vectors representing English words, we compute the n nearest vectors representing the n most similar Indonesian words. This is conventionally computed using the cosine of the angle between two vectors.

Within this general framework, there are several variables that we experiment with, as follows:

- **Collection size**. Three subsets of the parallel corpus were randomly created: P_{100} contains 100 article pairs, P_{500} contains 500 article pairs, and P_{1000} contains 1000 article pairs. Each subsequent subset wholly contains the previous subsets, i.e. $P_{100} \subset P_{500} \subset P_{1000}$.

[3] More specifically, the SQL version available from http://wnsqlbuilder.sourceforge.net

[4] publication forthcoming

[5] http://hantarto.definitionroadsafety.org

- **Rank reduction**. For each collection, we applied LSA with different degrees of rank approximation, namely 10%, 25%, and 50% the number of dimensions of the original collection. Thus, for P_{100} we compute the 10, 25, and 50-rank approximations, for P_{500} we compute the 50, 125, and 250-rank approximations, and for P_{1000} we compute the 100, 250, and 500-rank approximations.
- **Removal of stopwords**. Stopwords are words that appear numerously in a text, thus are assumed as insignificant to represent the specific context of the text. It is a common technique used to improve performance of information retrieval systems. It is applied in preprocessing the collections, i.e. removing all instances of the stopwords in the collections before applying LSA.
- **Weighting**. Two weighting schemes, namely TF-IDF and Log-Entropy, were applied to a word-document matrix separately.
- **Mapping Selection**. For computing the precision and recall values, we experimented with the number of mapping results to consider: the top 1, 10, 50, and 100 mappings based on similarity were taken.

film	0.814
filmnya	0.698
sutradara	0.684
garapan	0.581
perfilman	0.554
penayangan	0.544
kontroversial	0.526
koboi	0.482
irasional	0.482
frase	0.482

(a)

pembebanan	0.973
kijang	0.973
halmahera	0.973
alumina	0.973
terjadwal	0.973
viskositas	0.973
tabel	0.973
royalti	0.973
reklamasi	0.973
penyimpan	0.973

(b)

Table 2. The Most 10 Similar Indonesian Words for the English Words (a) Film and (b) Billion

As an example, Table 2 presents the results of mapping w^e_{film} and $w^e_{billion}$, i.e. the two English words *film* and *billion*, respectively to their Indonesian translations, using the P_{1000} training collection with 500-rank approximation. No weighting was applied. The former shows a successful mapping, while the latter shows an unsuccessful one. Bilingual LSA correctly maps w^e_{film} to its translation, w^i_{film}, despite the fact that they are treated as separate elements, i.e. their shared orthography is completely coincidental. Additionally, the other Indonesian words it suggests are semantically related, e.g. *sutradara* (director), *garapan* (creation), *penayangan* (screening), etc. On the other hand, the suggested word mappings for $w^e_{billion}$ are incorrect, and the correct translation, *milyar*, is missing. We suspect this may be due to several factors. Firstly, *billion* does not by itself invoke a particular semantic frame, and thus its semantic vector might not suggest a specific conceptual domain. Secondly, *billion* can sometimes be translated numerically instead of lexically. Lastly, this failure may also be due to the lack of data: the collection is simply too small to provide useful statistics that represent semantic context. Similar LSA approaches are commonly trained on collections of text numbering in the tens of thousands of articles.

Note as well that the absolute vector cosine values do not accurately reflect the correctness of the word translations. To properly assess the results of this experiment, evaluation against a gold standard is necessary. This is achieved by comparing its precision and recall against the Indonesian words returned by the bilingual dictionary, i.e. how isomorphic is the set of LSA-derived word mappings with a human-authored set of word mappings?

We provide a baseline as comparison, which computes the nearness between English and Indonesian words on the original word-document occurrence frequency matrix. Other approaches are possible, e.g. mutual information (Sari, 2007).

Table 3(a)-(e) shows the different aspects of our experiment results by averaging the other variables. Table 3(a) confirms our intuition that as the collection size increases, the precision and recall values also increase. Table 3(b) presents the effects of rank approximation. It shows that the higher the rank approximation percentage, the better the mapping results. Note that a rank approximation of 100% is equal to the FREQ baseline of simply using the full-rank word-document matrix for computing vector space nearness. Table 3(c) suggests that stopwords seem to help LSA to yield the correct mappings. It is believed that stopwords are not bounded by semantic domains, thus do not carry any semantic bias. However, on account of the small size of the collection, in coincidence, stopwords, which consistently appear in a specific domain, may carry some semantic information about the domain. Table 3(d) compares the mapping results in terms of weighting usage. It suggests that weighting can improve the mappings. Additionally, Log-Entropy weighting yields the highest results. Table 3(e) shows the comparison of mapping selections. As the number of translation pairs selected increases, the precision value decreases. On the other hand, as the number of translation pairs selected increases, the possibility to find more pairs matching the pairs in bilingual dictionary increases. Thus, the recall value increases as well.

Collection Size	FREQ		LSA	
	P	R	P	R
P_{100}	0.0668	0.1840	0.0346	0.1053
P_{500}	0.1301	0.2761	0.0974	0.2368
P_{1000}	**0.1467**	**0.2857**	0.1172	0.2603

(a)

Rank Approximation	P	R
10%	0.0680	0.1727
25%	0.0845	0.2070
50%	0.0967	0.2226
100%	**0.1009**	**0.2285**

(b)

Stopwords	FREQ		LSA	
	P	R	P	R
Contained	0.1108	**0.2465**	0.0840	0.2051
Removed	**0.1138**	0.2440	0.0822	0.1964

(c)

Weighting Usage	FREQ		LSA	
	P	R	P	R
No Weighting	0.1009	0.2285	0.0757	0.1948
Log-Entropy	**0.1347**	**0.2753**	0.1041	0.2274
TF-IDF	0.1013	0.2319	0.0694	0.1802

(d)

Mapping Selection	FREQ		LSA	
	P	R	P	R
Top 1	**0.3758**	0.1588	0.2380	0.0987
Top 10	0.0567	0.2263	0.0434	0.1733
Top 50	0.0163	0.2911	0.0133	0.2338
Top 100	0.0094	**0.3183**	0.0081	0.2732

(e)

Table 3. Results of bilingual word mapping comparing (a) collection size, (b) rank approximation, (c) removal of stopwords, (d) weighting schemes, and (e) mapping selection

Most interestingly, however, is the fact that the FREQ baseline, which uses the basic vector space model, consistently outperforms LSA.

4.3 Bilingual Concept Mapping

Using the same resources from the previous experiment, we ran an experiment to perform bilingual concept mapping by replacing the vectors to be compared with semantic vectors for concepts (see Section 3). For concept $c^e \in C^e$, i.e. a WordNet synset, we constructed a set of textual context as the union of $\omega(c)$, the set of words in the gloss of c^e, and the set of words in the example sentences associated with c^e. To represent our intuition that the words in $\omega(c)$ played more of an important role in defining the semantic vector than the words in the gloss and example, we applied a weight of 60%, 30%, and 10% to the three components, respectively. Similarly, a semantic vector representing a concept $c^i \in C^i$, i.e. an Indonesian word sense in the KBBI, was constructed from a textual context set composed of the sublemma, the definition, and the example of the word sense, using the same weightings. We only average word vectors if they appear in the collection (depending on the experimental variables used).

We formulated an experiment which closely resembles the word sense disambiguation problem: given a WordNet synset, the task is to select the most appropriate Indonesian sense from a subset of senses that have been selected based on their words appearing in our bilingual dictionary. These specific senses are called suggestions. Thus, instead of comparing the vector representing *communication* with every single Indonesian sense in the KBBI, in this task we only compare it against suggestions with a limited range of sublemmas, e.g. *komunikasi*, *perhubungan*, *hubungan*, etc.

This setup is thus identical to that of an ongoing experiment here to manually map WordNet synsets to KBBI senses. Consequently, this facilitates assessment of the results by computing the level of agreement between the LSA-based mappings with human annotations.

To illustrate, Table 4(a) and 4(b) presents a successful and unsuccessful example of mapping a WordNet synset. For each example we show the synset ID and the ideal textual context set, i.e. the set of words that convey the synset, its gloss and example sentences. We then show the actual textual context set with the notation {{X}, {Y}, {Z}}, where X, Y , and Z are the subset of words that appear in the training collection. We then show the Indonesian word sense deemed to be most similar. For each sense we show the vector similarity score, the KBBI ID and its ideal textual context set, i.e. the sublemma, its definition and example sentences. We then show the actual textual context set with the same notation as above.

WordNet Synset ID: 100319939, **Words**: chase, following, pursual, pursuit, **Gloss**: the act of pursuing in an effort to overtake or capture, **Example**: the culprit started to run and the cop took off in pursuit, **Textual context set**: {{following, chase}, {the, effort, of, to, or, capture, in, act, pursuing, an}, {the, off, took, to, run, in, culprit, started, and}}

KBBI ID: k39607 - **Similarity**: 0.804, **Sublemma**: mengejar, **Definition**: berlari untuk menyusul menangkap dsb memburu, **Example**: ia berusaha mengejar dan menangkap saya, **Textual context set**: {{mengejar}, {memburu, berlari, menangkap, untuk, menyusul},{berusaha, dan, ia, mengejar, saya, menangkap}}

(a)

WordNet synset ID: 201277784, **Words**: crease, furrow, wrinkle
Gloss: make wrinkled or creased, **Example**: furrow one's brow,
Textual context set: {{}, {or, make}, {s, one}}

KBBI ID: k02421 - **Similarity**: 0.69, **Sublemma**: alur, **Definition**: jalinan peristiwa dl karya sastra untuk mencapai efek tertentu pautannya dapat diwujudkan oleh hubungan temporal atau waktu dan oleh hubungan kausal atau sebab-akibat, **Example**: (none), **Textual context set**: {{alur}, {oleh, dan, atau, jalinan, peristiwa, diwujudkan, efek, dapat, karya, hubungan, waktu, mencapai, untuk, tertentu}, {}}

(b)

Table 4. Example of (a) Successful and (b) Unsuccessful Concept Mappings

In the first example, the textual context sets from both the WordNet synset and the KBBI senses are fairly large, and provide sufficient context for LSA to choose the correct KBBI sense. However, in the second example, the textual context set for the synset is very small, due to the words not appearing in the training collection. Furthermore, it does not contain any of the words that truly convey the concept. As a result, LSA is unable to identify the correct KBBI sense.

For this experiment, we used the P_{1000} training collection. The results are presented in Table 5. As a baseline, we select three random suggested Indonesian word senses as a mapping for an English word sense. The reported random baseline in Table 5 is an average of 10 separate runs. Another baseline was computed by comparing English common-based concepts to their suggestion based on a full rank word-document matrix. Top 3 Indonesian concepts with the highest similarity values are designated as the mapping results. Subsequently, we compute the Fleiss kappa (Fleiss, 1971) of this result together with the human judgements.

The average level of agreement between the LSA mappings 10% and the human judges (0.2713) is not as high as between the human judges themselves (0.4831). Nevertheless, in general it is better than the random baseline (0.2380) and frequency baseline (0.2132), which suggests that LSA is indeed managing to capture some measure of bilingual semantic information implicit within the parallel corpus.

Furthermore, LSA mappings with 10% rank approximation yields higher levels of agreement than LSA with other rank approximations. It is contradictory with the word mapping results where LSA with bigger rank approximations yields higher results (Section 4.2).

5 Discussion

Previous works have shown LSA to contribute positive gains to similar tasks such as Cross Language Information Retrieval (Rehder et al., 1997). However, the bilingual word mapping results presented in Section 4.3 show the basic vector space model consistently outperforming LSA at that particular task, despite our initial intuition that LSA should actually improve precision and recall.

We speculate that the task of bilingual word mapping may be even harder for LSA than that of

Judges	Synsets	Fleiss Kappa Values					
		Judges only	Judges + RNDM3	Judges + FREQ Top 3	Judges + LSA 10% Top3	Judges + LSA 25% Top3	Judges + LSA 50% Top3
≥ 2	144	0.4269	0.1318	0.1667	0.1544	0.1606	0.1620
≥ 3	24	0.4651	0.2197	0.2282	0.2334	0.2239	0.2185
≥ 4	8	0.5765	0.3103	0.2282	0.3615	0.3329	0.3329
≥ 5	4	0.4639	0.2900	0.2297	0.3359	0.3359	0.3359
Average		0.4831	0.2380	0.2132	0.2713	0.2633	0.2623

Table 5. Results of Concept Mapping

bilingual concept mapping due to its finer alignment granularity. While concept mapping attempts to map a concept conveyed by a group of semantically related words, word mapping attempts to map a word with a specific meaning to its translation in another language.

In theory, LSA employs rank reduction to remove noise and to reveal underlying information contained in a corpus. LSA has a 'smoothing' effect on the matrix, which is useful to discover general patterns, e.g. clustering documents by semantic domain. Our experiment results, however, generally shows the frequency baseline, which employs the full rank word-document matrix, outperforming LSA.

We speculate that the rank reduction perhaps blurs some crucial details necessary for word mapping. The frequency baseline seems to encode more cooccurrence than LSA. It compares word vectors between English and Indonesian that contain pure frequency of word occurrence in each document. On the other hand, LSA encodes more semantic relatedness. It compares English and Indonesian word vectors containing estimates of word frequency in documents according to the context meaning. Since the purpose of bilingual word mapping is to obtain proper translations for an English word, it may be better explained as an issue of cooccurrence rather than semantic relatedness. That is, the higher the rate of cooccurrence between an English and an Indonesian word, the likelier they are to be translations of each other.

LSA may yield better results in the case of finding words with similar semantic domains. Thus, the LSA mapping results should be better assessed using a resource listing semantically related terms, rather than using a bilingual dictionary listing translation pairs. A bilingual dictionary demands more specific constraints than semantic relatedness, as it specifies that the mapping results should be the translations of an English word.

Furthermore, polysemous terms may become another problem for LSA. By rank approximation, LSA estimates the occurrence frequency of a word in a particular document. Since polysemy of English terms and Indonesian terms can be quite different, the estimations for words which are mutual translations can be different. For instance, *kali* and *waktu* are Indonesian translations for the English word *time*. However, *kali* is also the Indonesian translation for the English word *river*. Suppose *kali* and *time* appear frequently in documents about multiplication, but *kali* and *river* appear rarely in documents about river. Then, *waktu* and *time* appear frequently in documents about time. As a result, LSA may estimate *kali* with greater frequency in documents about multiplication and time, but with lower frequency in documents about river. The word vectors between *kali* and *river* may not be similar. Thus, in bilingual word mapping, LSA may not suggest *kali* as the proper translation for *river*. Although polysemous words can also be a problem for the frequency baseline, it merely uses raw word frequency vectors, the problem does not affect other word vectors. LSA, on the other hand, exacerbates this problem by taking it into account in estimating other word frequencies.

6 Summary

We have presented a model of computing bilingual word and concept mappings between two semantic lexicons, in our case Princeton WordNet and the KBBI, using an extension to LSA that exploits implicit semantic information contained within a parallel corpus.

The results, whilst far from conclusive, indicate that that for bilingual word mapping, LSA is consistently outperformed by the basic vector space model, i.e. where the full-rank word-document matrix is applied, whereas for bilingual concept mapping LSA seems to slightly improve results. We speculate that this is due to the fact that LSA, whilst very useful for revealing implicit semantic patterns, blurs the cooccurrence information that is necessary for establishing word translations.

We suggest that, particularly for bilingual word mapping, a finer granularity of alignment, e.g. at the sentential level, may increase accuracy (Deng and Gao, 2007).

Acknowledgment

The work presented in this paper is supported by an RUUI (*Riset Unggulan Universitas Indonesia*) 2007 research grant from DRPM UI (*Direktorat Riset dan Pengabdian Masyarakat Universitas Indonesia*). We would also like to thank Franky for help in software implementation and Desmond Darma Putra for help in computing the Fleiss kappa values in Section 4.3.

References

Eneko Agirre and Philip Edmonds, editors. 2007. *Word Sense Disambiguation: Algorithms and Applications.* Springer.

Katri A. Clodfelder. 2003. *An LSA Implementation Against Parallel Texts in French and English.* In Proceedings of the HLT-NAACL Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, 111–114.

Yonggang Deng and Yuqing Gao. June 2007. *Guiding statistical word alignment models with prior knowledge.* In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 1–8, Prague, Czech Republic. Association for Computational Linguistics.

Christiane Fellbaum, editor. May 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Joseph L. Fleiss. 1971.*Measuring nominal scale agreement among many raters.* Psychological Bulletin, 76(5):378–382.

Dan Kalman. 1996. *A singularly valuable decomposition: The svd of a matrix.* The College Mathematics Journal, 27(1):2–23.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. *An introduction to latent semantic analysis.* Discourse Processes, 25:259–284.

Bob Rehder, Michael L. Littman, Susan T. Dumais, and Thomas K. Landauer. 1997. *Automatic 3-language cross-language information retrieval with latent semantic indexing.* In Proceedings of the Sixth Text Retrieval Conference (TREC-6), pages 233–239.

Syandra Sari. 2007. *Perolehan informasi lintas bahasa indonesia-inggris berdasarkan korpus paralel dengan menggunakan metoda mutual information dan metoda similarity thesaurus.* Master's thesis, Faculty of Computer Science, University of Indonesia, Call number: T-0617.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 97-105

Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition

Tara McIntosh and **James R. Curran**
School of IT
University of Sydney
NSW 2006, Australia
{tara,james}@it.usyd.edu.au

Abstract

We present the *Weighted Mutual Exclusion Bootstrapping* (WMEB) algorithm for simultaneously extracting precise semantic lexicons and templates for multiple categories. WMEB is capable of extracting larger lexicons with higher precision than previous techniques, successfully reducing semantic drift by incorporating new weighting functions and a cumulative template pool while still enforcing mutual exclusion between the categories.

We compare WMEB and two state-of-the-art approaches on the Web 1T corpus and two large biomedical literature collections. WMEB is more efficient and scalable, and we demonstrate that it significantly outperforms the other approaches on the noisy web corpus and biomedical text.

1 Introduction

Automatically acquiring semantic lexicons and templates from raw text is essential for overcoming the knowledge bottleneck in many natural language processing tasks, e.g. question answering (Ravichandran and Hovy, 2002). These tasks typically involve identifying named entity (NE) classes which are not found in annotated corpora and thus supervised NE recognition models are not always available. This issue becomes even more evident in new domains, such as biomedicine, where new semantic categories are often poorly represented in linguistic resources, if at all (Hersh et al., 2007).

There are two common approaches to extract semantic lexicons: distributional similarity and template-based bootstrapping . In *template-based bootstrapping* algorithms, templates that express a particular semantic type are used to recognise new terms, and in turn these new terms help identify new templates iteratively (Riloff and Jones, 1999). These algorithms are attractive as they are domain and language independent, require minimal linguistic preprocessing, are relatively efficient, and can be applied to raw text.

Unfortunately, *semantic drift* often occurs when ambiguous or erroneous terms or patterns are introduced into the lexicon or set of templates. Curran et al. (2007) developed Mutual Exclusion Bootstrapping (MEB) to reduce semantic drift by forcing semantic classes to be mutually exclusive.

We introduce a new algorithm, Weighted Mutual Exclusion Bootstrapping (WMEB), that automatically acquires multiple semantic lexicons and their templates simultaneously. It extends on the Curran et al. (2007) assumption of mutual exclusion between categories by incorporating a novel cumulative template pool and new term and template weighting functions.

We compare WMEB against two state-of-the-art mutual bootstrapping algorithms, MEB (Curran et al., 2007) and BASILISK (Thelen and Riloff, 2002). We have evaluated the terms and templates these algorithms extract under a range of conditions from three raw text collections: noisy web text, biomedical abstracts, and full-text articles.

We demonstrate that WMEB outperforms these existing algorithms in extracting precise lexicons and templates from all three datasets. WMEB is significantly less susceptible to semantic drift and so can produce large lexicons accurately and efficiently across multiple domains.

2 Background

Hearst (1992) pioneered the use of templates for information extraction, focussing on acquiring *is-a* relations using manually devised templates like *such W as X, ..., Y and/or Z* where *X, ..., Y, Z* are hyponyms of *W*. Various automated template-based bootstrapping algorithms have since been developed to iteratively build semantic lexicons from texts. Riloff and Shepherd (1997) proposed *Iterative Bootstrapping* (IB) where seed instances of a semantic category are used to identify related terms that frequently co-occur.

In *Mutual Bootstrapping* (MB) (Riloff and Jones, 1999) seed instances of a desired type are used to infer new templates, which in turn identify new lexicon entries. This process is repeated with the new terms identifying new templates. In each iteration, new terms and templates are selected based on a metric scoring their suitability for extracting additional templates and terms for the category. Unfortunately, if a term with multiple senses or a template which weakly constrains the semantic class is selected, *semantic drift* of the lexicon and templates occurs – the semantic class drifts into another category (Curran et al., 2007).

Extracting multiple semantic categories simultaneously has been proposed to reduce semantic drift. The bootstrapping instances compete with one another in an attempt to actively direct the categories away from each other (Thelen and Riloff, 2002; Yangarber et al., 2002; Curran et al., 2007). This strategy is similar to the one sense per discourse assumption (Yarowsky, 1995).

In BASILISK (Thelen and Riloff, 2002), candidate terms for a category are ranked highly if they have strong evidence for the category and little or no evidence for another. It is possible for an ambiguous term to be assigned to the less dominant sense, and in turn less precise templates will be selected, causing semantic drift. Drift may also be introduced as templates can be selected by different categories in different iterations.

NOMEN (Yangarber et al., 2002) was developed to extract generalized names such as diseases and drugs, with no capitalisation cues. NOMEN, like BASILISK, identifies semantic category lexicons in parallel, however NOMEN extracts the left and right contexts of terms independently and generalises the contexts.

Curran et al. (2007) introduced the algorithm *Mutual Exclusion Bootstrapping* (MEB) which more actively defines the semantic boundaries of the lexicons extracted simultaneously. In MEB, the categories compete for both terms and templates. Semantic drift is reduced in two ways: by eliminating templates that collide with two or more categories in an iteration (from all subsequent iterations), and by ignoring colliding candidate terms (for an iteration). This effectively excludes general templates that can occur frequently with multiple categories, and reduces the chance of assigning ambiguous terms to their less dominant sense.

The scoring metric for candidate terms and templates in MEB is simple and naïve. Terms and templates which 1) match the most input instances, and 2) have the potential to generate the most new candidates, are preferred (Curran et al., 2007). This second criteria aims to increase recall, however the selected instances are highly likely to introduce drift. We introduce a new weighting scheme to effectively overcome this.

Template-based bootstrapping algorithms have also been used in various Information Extraction (IE) tasks. Agichtein and Gravano (2000) developed the SNOWBALL system to identify the locations of companies, and Yu and Agichtein (2003) applied SNOWBALL to extract synonymous gene and protein terms. Pantel and Pennacchiotti (2006) used bootstrapping to identify numerous semantic relationships, such as *is-a* and *part-of* relationships. They incorporate the *pointwise mutual information* (MI) measure between the templates and instances to determine template reliability, as well as exploiting generic templates and the Web for filtering incorrect instances. We evaluate the effectiveness of MI as a weighting function for selecting terms and templates in WMEB.

In the biomedical domain, there is an increased interest in automatically extracting lexicons of biomedical entities such as *antibodies* and *mutations*, and the templates which extract such terms. This is primarily due to the lack, and scope, of annotated resources, and the introduction of new semantic categories which are severely under-represented in corpora and lexicons. Meij and Katrenko (2007) applied MB to identify biomedical entities and their templates, which were both then used to find potential answer sentences for the TREC Genomics Track task (Hersh et al., 2007). The accuracy of their extraction process was not evaluated, however their Information Retrieval system had performance gains in unambiguous and common entity types, where little semantic drift is likely to occur.

3 Weighted MEB (WMEB) Algorithm

Our algorithm, *Weighted Mutual Exclusion Bootstrapping* (WMEB), extends MEB described in Curran et al. (2007). MEB is a minimally supervised, mutual bootstrapping algorithm which reduces semantic drift by extracting multiple semantic categories with individual bootstrapping instances in parallel, and by forcing the categories to be mutually exclusive (Figure 1). In MEB, the templates describe the context of a term (two terms to the left and right). Each MEB instance iterates simultaneously between two stages: template extraction and selection, and term extraction and selection. The key assumption of MEB is that terms only have a single sense and that templates only extract terms of a single sense. This is forced by excluding terms and templates from all categories if in one iteration they are selected by more than one category.

In this section we describe the architecture of WMEB. WMEB employs a new weighting scheme, which identifies candidate templates and terms that are strongly associated with the lexicon terms and their templates respectively. In WMEB, we also introduce the concept of a cumulative template pool. These techniques reduce the semantic drift in WMEB more effectively than in MEB.

3.1 System Architecture

WMEB takes as input a set of manually labelled seed terms for each category. Each category's seed set forms it's initial lexicon.

Template Extraction and Selection

For each term in the category lexicon, WMEB extracts all candidate templates the term matches. To enforce mutually exclusive templates, candidate templates identified by multiple categories are excluded from the candidate set and all subsequent iterations. The remaining candidates are then ranked according to their *reliability* measure and their *relevance weight* (see Section 3.2).

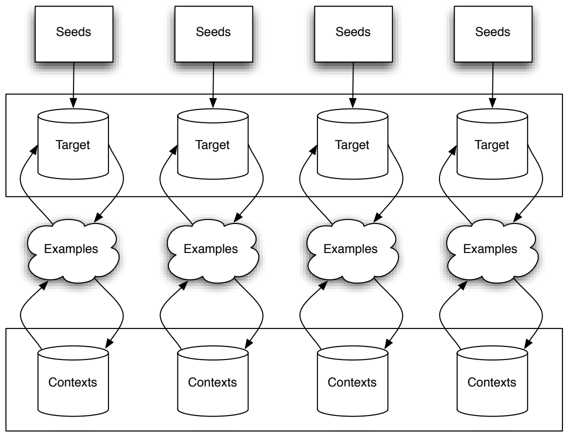


Figure 1: MEB architecture.

After manually inspecting the templates selected by MEB and BASILISK, we introduced the *cumulative template pool* (pool) in WMEB. In MEB (Curran et al., 2007) and BASILISK (Thelen and Riloff, 2002), the top-k[1] templates for each iteration are used to extract new candidate terms. We observed that as the lexicons grow, more general templates can drift into the top-k. This was also noted by Jones et al. (1999). As a result the earlier precise templates lose their influence.

WMEB successfully overcomes this by accumulating all selected templates from the current and all previous iterations in the pool, ensuring previous templates can contribute. The templates in the pool have equal weight in all iterations.

In WMEB, the top-k templates are selected for addition to the pool. If all top-k templates are already in the pool, then the next available top template is added. This ensures at least one new template is added in each iteration.

Term Extraction and Selection

For each template in a category's pool, all available candidate terms matching the templates are identified. Like the candidate templates, terms which are extracted by multiple categories are also excluded. A colliding term will collide in all consecutive iterations due to the cumulative pool and thus WMEB creates a stricter term boundary between categories than MEB. The candidate

[1] BASILISK also adds an additional template in each iteration, i.e. k is increased by one in each iteration.

terms are ranked with respect to their *reliability* and *relevance weight*, and the top-k terms are added to the category's lexicon.

3.2 Term and Template Weighting

In MEB, candidate terms and templates are ranked according to their *reliability* measure and ties are broken using the *productivity* measure. The *reliability* of a term for a given category, is the number of input templates in an iteration that can extract the term. The *productivity* of a term is the number of potentially new templates it may add in the next iteration. These measures are symmetrical for both terms and templates. More reliable instances would theoretically have higher precision, while high productive instances will have a high recall. Unfortunately, high productive instances could potentially introduce drift.

In WMEB we replace the productivity measure with a new *relevance weight*. We have investigated scoring metrics which prefer terms and templates that are highly associated with their input instances, including: the chi-squared (χ^2) statistic and three variations of the pointwise mutual information (MI) measure (Manning and Schutze, 1999, Chapter 5). Each of these estimates the strength of the co-occurance of a term and a template. They do not give the likelihood of the instance being a member of a semantic category.

The first variation of MI we investigate is MI^2, which scales the probability of the term (t) and template (c) pair to ensure more frequent combinations have a greater weight.

$$\text{MI}^2(t,c) = \log_2 \frac{p(t,c)^2}{p(t)p(c)}$$

Each of the probabilities are calculated directly from the relative frequencies without smoothing. The scores are set to 0 if their observed frequencies are less than 5, as these estimates are sensitive to low frequencies.

The other variation of MI function we utilise is truncated MI (MIT), and is defined as:

$$\text{MIT}(t,c) = \begin{cases} \text{MI}^2(t,c) & : \text{MI}(t,c) > 0 \\ 0 & : \text{MI}(t,c) \leq 0 \end{cases}$$

The overall *relevance weight* for a term or template is the sum of the scores of the pairs, where score corresponds to one of the scoring metrics, and C is the set of templates matching term t, and T is the set of terms matching template c.

TYPE (#)	MEDLINE	TREC	Web1T
Terms	1 347 002	1 478 119	568 202
Templates	4 090 412	8 720 839	10 568 219
5-grams	72 796 760	63 425 523	42 704 392
Orig. tokens	6 642 802 776	3 479 412 905	~ 1 trillion

Table 1: Filtered 5-gram dataset statistics.

$$\text{weight}(t) = \sum_{c \in C} \text{score}(t,c)$$
$$\text{weight}(c) = \sum_{t \in T} \text{score}(c,t)$$

The terms and templates are ordered by their *reliability*, and ties are broken by their *relevance weight*. WMEB is much more efficient than BASILISK using these weighting scores – for all possible term and template pairs, the scores can be pre-calculated when the data is loaded, whereas in BASILISK, the scoring metric is more computationally expensive. In BASILISK, each individual calculation is dependent on the current state of the bootstrapping process, and therefore scores cannot be pre-calculated.

4 Experimental Setting

4.1 Data

We evaluated the performance of BASILISK, MEB and WMEB using 5-grams from three raw text resources: the Google Web 1T corpus (Brants and Franz, 2006), MEDLINE abstracts[2] and the TREC Genomics Track 2007 full-text articles (Hersh et al., 2007). In our experiments, the *term* is the middle token of each 5-gram and the *template* is the two tokens on either side. Unlike Riloff and Jones (1999) and Yangarber (2003), we do not use syntactic knowledge. Although we only extract unigrams, each algorithm can identify multi-term entities (Murphy and Curran, 2007).

The Web 1T 5-grams were filtered by removing templates appearing with only one term and templates containing numbers. All 5-gram contexts with a non-titlecase term were also filtered as we are extracting proper nouns.

[2]The set contains all MEDLINE abstracts available up to Oct 2007 (16 140 000 abstracts)

CAT	DESCRIPTION
FEM	Person: female first name *Mary Patricia Linda Barbara Elizabeth*
MALE	Person: male first name *James John Robert Michael William*
LAST	Person: last name *Smith Johnson Williams Jones Brown*
TTL	Honorific title *General President Director King Doctor*
NORP	Nationality, Religion, Political (adjectival) *American European French British Western*
FOG	Facilities and Organisations *Ford Microsoft Sony Disneyland Google*
PLCE	Place: Geo-political entities and locations *Africa America Washington London Pacific*
DAT	Reference to a date or period *January May December October June*
LANG	Any named language *English Chinese Japanese Spanish Russian*

Table 2: Web 1T semantic categories and seeds.

Limited preprocessing was required to extract the 5-grams from MEDLINE and TREC. The TREC documents were converted from HTML to raw text, and both collections were tokenised using bio-specific NLP tools (Grover et al., 2006). We did not exclude lowercase terms or templates containing numbers. Templates appearing with less than 7 (MEDLINE) or 3 (TREC) terms were removed. These frequencies were selected to permit the largest number of templates and terms loadable by BASILISK [3], to allow a fair comparison.

The size of the resulting datasets are shown in Table 1. Note that, Web 1T has far fewer terms but many more templates than the biomedical sets, and TREC articles result in more templates than MEDLINE for a similar number of terms.

4.2 Semantic Categories & Stop Categories

In the Web 1T experiments, we are extracting proper-noun NE and their templates. We use the categories from Curran et al. (2007) which are a subset of those in the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005), and are shown in Table 2.

In the MEDLINE and TREC experiments we considered the TREC Genomics 2007 entities with a few modifications (Hersh et al., 2007). We excluded the categories *Toxicities*, *Pathways* and

[3]BASILISK requires n times more memory to store the term and template frequency counts than MEB and WMEB, where n is the number of categories.

CAT	DESCRIPTION
ANTI	Antibodies: Immunoglobulin molecules *MAb IgG IgM rituximab infliximab*
CELL	Cells: A morphological or functional form of a cell *RBC HUVEC BAEC VSMC SMC*
CLNE	Cell lines: Cell clones grown in tissue culture *PC12 CHO HeLa Jurkat COS*
DISE	Diseases: pathological process affecting organisms *asthma hepatitis tuberculosis HIV malaria*
DRUG	Drugs: A pharmaceutical preparation *acetylcholine carbachol heparin penicillin tetracyclin*
FUNC	Molecular functions and processes *kinase ligase acetyltransferase helicase binding*
MUTN	Gene and protein mutations, and mutants *Leiden C677T C28Y L5178Y S100B* (MEDLINE) *T47D S100B K44A F442A G93A* (TREC)
PROT	Proteins and genes *p53 actin collagen albumin IL-6*
SIGN	Signs and symptoms of diseases *anemia hypertension hyperglycemia fever cough*
TUMR	Tumor types *lymphoma sarcoma melanoma neuroblastoma osteosarcoma*

Table 3: Biomedical semantic categories and seeds.

Biological Substances, which are predominately multi-term entities, and the category *Strains* due to the difficulty for biologists to distinguish between strains and organisms. We combined the categories *Genes* and *Proteins* into PROT as there is a very high degree of metonymy between these, particularly once out of context. We were also interested in the fine grain distinction between types of *cells* and *cell lines*, so we split the *Cell or Tissue Type* category into CELL and CLNE entities.

Five seed terms (as non-ambiguous as possible) were selected for each category based on the evaluators' knowledge of them and their high frequency counts in the collections, and are shown in Table 2 and 3. Separate MUTN seeds for MEDLINE and TREC were used as some high frequency MUTN terms in MEDLINE do not appear in TREC.

As in Curran et al. (2007) and Yangarber (2003), we used additional *stop categories*, which are extracted as well but then discarded. Stop categories help constrain the categories of interest by creating extra boundaries against semantic drift. For the Web 1T experiments we used the stop categories described in Curran et al. (2007) – ADDRESS, BODY, CHEMICAL, COLOUR, DRINK, FOOD, JEWEL and WEB terms. In the biomedical experiments we introduced four stop categories – AMINO ACID, ANIMAL, BODY and ORGANISM.

4.3 Evaluation

Our evaluation process involved manually inspecting each extracted term and judging whether it was a member of the semantic class. The biomedical terms were evaluated by a domain expert. Unfamiliar terms were checked using online resources including MEDLINE, Medical Subject Headings (MeSH), Wikipedia and Google.

Each ambiguous term was counted as correct if it was classified into one of its correct categories. If a single term was unambiguously part of a multi-word term we considered it correct. Modifiers such as *cultured* in *cultured lymphocytes* and *chronic* in *chronic arthritis* were marked as incorrect.

For comparing the accuracy of the systems we evaluated the precision of the first *n* selected terms for each category. In some experiments we report the average precision over each category (Av(*n*)).

Our evaluation also includes judging the quality of the templates extracted. This is the first empirical evaluation of the templates identified by bootstrapping algorithms. We inspected the first 100 templates extracted for each category, and classified them into three groups. Templates where the context is semantically coherent with terms only from the assigned category are considered to accurately define the category and are classified as *true matches* (TM). Templates where another category's term could also be inserted were designated as *possible matches* (PM). For example, DRUG matches: *pharmacokinetics of* X *in patients* and *mechanism of* X *action* ., however the latter is a PM as it also matches PROT. Templates like *compared with* X *for the* and *Value of* X *in the* are *false matches* as they do not provide any contextual information for a specific entity.

5 Results

All of our experiments use the stop categories mentioned in §4.3, unless otherwise stated. The maximum number of terms and templates (top-k) which can be added in each iteration is 5.

Our first experiment investigates the weighting functions for WMEB on the Web 1T and MEDLINE data (Table 4). For the Web 1T data, all of the measures are approximately equal at 400 extracted terms. On MEDLINE, χ^2 outperforms the other measures and is more consistent. In Table 4 we also see the first difference between the two domains. The MEDLINE data scores are significantly higher, with little semantic drift down to 400 terms. For the remaining WMEB experiments we use the χ^2 weighting function.

	Web 1T				MEDLINE			
	100	200	300	400	100	200	300	400
χ^2	78.5	74.2	69.1	65.2	87.1	89.5	89.0	89.4
MI	76.0	70.7	67.4	65.0	84.7	87.0	86.8	87.2
MI^2	80.7	72.7	68.1	64.7	84.3	82.8	82.3	80.7
MIT	79.3	74.4	69.1	65.5	86.1	84.4	84.3	83.8

Table 4: Results comparing WMEB scoring functions.

5.1 Terms

Table 5 and 6 summarise the comparison of BASILISK, MEB and WMEB, on the Web 1T and MEDLINE data respectively. The category analysis is measured on the top 100 terms. BASILISK outperforms MEB on both datasets, whereas WMEB performs similarly to BASILISK on the Web 1T data. For the MEDLINE data, WMEB outperforms both BASILISK and MEB.

Each algorithm will stop extracting terms in a category if all candidate terms are exhausted. Templates may also become exhausted. Thus, each algorithm may be penalised when we evaluate past their stopping points. We have provided adjusted scores in brackets to take this into account. After adjustment, WMEB and BASILISK significantly outperform MEB, and WMEB is more accurate than BASILISK.

It is clear that some categories are much easier than others to extract, e.g. LAST and CELL, while others are quite difficult, e.g. NORP and FUNC. For many categories there is a wide variation across the algorithms' performance, e.g. NORP and TUMR.

The stop categories' seeds were optimised for MEB. When the stop categories are introduced BASILISK gains very little compared to MEB and WMEB. In BASILISK-NOSTOP categories rarely drift into unspecified categories, however they do drift into similar semantic categories of interest, e.g. TUMR drifts into CLNE and vice-versa, in early iterations. This is because BASILISK weakly defines the semantic boundaries. In WMEB, this rarely occurs as the boundaries are strict. We find DISE drifts into BODY and thus a significant per-

CAT	NO STOP			STOP		
	BAS	MEB	WMEB	BAS	MEB	WMEB
FEM	98	100	100	99	100	100
MALE	97	100	100	97	100	98
LAST	100	100	100	100	100	100
TTL	39	9	46	37	31	53
NORP	67	17	41	64	22	40
FOG	90	98	98	90	95	98
PLCE	98	100	94	96	100	98
DATE	35	47	24	38	57	23
LANG	97	90	96	95	93	97
Av(100)	**80.2**	73.4	77.7	**79.5**	77.6	78.6
	(84.7)		(**86.1**)	(84.3)		(**87.1**)
Av(200)	73.5	68.7	**71.0**	72.2	73.2	**74.2**
	(**84.5**)		(80.8)	(82.4)		(**85.4**)

Table 5: Results comparing Web 1T terms.

CAT	NO STOP			STOP		
	BAS	MEB	WMEB	BAS	MEB	WMEB
ANTI	47	95	98	49	92	96
CELL	95	95	98	95	98	100
CLNE	91	93	96	81	100	100
DISE	77	33	49	82	39	76
DRUG	67	77	92	69	92	100
FUNC	73	60	71	73	61	81
MUTN	88	87	87	88	63	81
PROT	99	99	100	99	100	99
SIGN	96	55	67	97	95	99
TUMR	51	33	39	51	23	39
Av(100)	78.5	72.7	**79.7**	78.4	76.3	**87.1**
Av(200)	75.0	66.1	78.6	74.7	70.1	89.5
Av(300)	72.3	60.2	78.3	72.1	64.8	89.0
Av(400)	70.0	56.3	**77.4**	71.0	60.8	**89.4**

Table 6: Results comparing MEDLINE terms.

formance gain is achieved with the BODY stop category.

The remainder of the analysis will be performed on the biomedical data. In Table 7, we can observe the degree of drift which occurs in each algorithm on MEDLINE for a given number of extracted terms. For example, the 101–200 row gives the accuracy of the 101–200th extracted terms. WMEB performs fairly consistently in each range, whereas MEB degrades quickly and BASILISK to a lesser extent. The terms extracted by WMEB in later stages are more accurate than the first 100 extracted terms identified by BASILISK and MEB.

In practice, it is unlikely we would only have 5 seed terms. Thus, we investigated the impact of using 100 seeds as input for each algorithm (Table 8). Only BASILISK improved with these large

RANGE	BAS	MEB	WMEB
0–100	78.4	76.3	87.1
101–200	71.0	63.8	91.8
201–300	66.9	54.3	88.0
301–400	67.6	48.7	90.7
401–500	69.5	49.7	83.5

Table 7: Drift results on MEDLINE.

AV(N)	BAS	MEB	WMEB
50	82.4	62.4	72.0
100	83.4	57.0	71.3
200	**82.7**	53.9	72.2
5 seeds: 200	78.4	**76.3**	**89.5**

Table 8: Results comparing 100 seeds on MEDLINE.

seed sets, however it did not outperform WMEB with only 5 input seeds. WMEB and MEB do not gain from these additional seeds as they severely limit the search space by introducing many more colliding templates in the early iterations.

5.2 Templates

Previous work has not evaluated the quality of the templates extracted, which is crucial for tasks like question answering that will utilise the templates. Our evaluation compares the first 100 templates identified by each algorithm. Table 9 shows the distribution of *true* and *possible matches*.

Although each algorithm performs well on CELL and CLNE, the templates for these are predominately PM. This is due to the difficulty of disambiguating CELL from CLNE. Categories which are hard to identify have far more partial and false matches. For example, the majority of TUMR templates can also semantically identify BODY. WMEB still performs well on categories with few TM, in particular SIGN (99% with 54 PM) . This is a result of mutual exclusion which forces those templates to a single category.

5.3 Other experiments

BASILISK is noticeably less efficient than MEB (14 times slower on Web 1T, 5 times on MEDLINE) and WMEB (10 times slower on Web 1T, 4 times on MEDLINE). BASILISK cannot precalculate the scoring metrics as they are dependent on the state of the bootstrapping process.

Table 10 shows the effectiveness of WMEB's individual components on MEDLINE. Here MEB is

CAT	BAS		MEB		WMEB	
	TM	PM	TM	PM	TM	PM
ANTI	63	8	97	0	**100**	0
CELL	2	**98**	1	**68**	2	**84**
CLNE	1	99	79	21	78	22
DISE	80	15	5	81	95	5
DRUG	80	17	82	16	78	17
FUNC	62	33	10	42	49	50
MUTN	3	27	3	26	9	91
PROT	98	1	54	0	99	0
SIGN	93	6	90	5	12	**54**
TUMR	4	94	0	81	2	67
Av(100)	48.6	39.8	42.1	34.0	**52.4**	39.0

Table 9: Results comparing MEDLINE templates

	100	200	500
MEB	76.3	70.0	58.6
WMEB-pool	83.2	81.7	77.8
WMEB-weight	82.3	79.5	76.4
WMEB	87.1	89.5	88.2

Table 10: Effect of WMEB weights and pool.

the baseline with no pool or weighting. WMEB-pool corresponds to WMEB with weighting and without the cumulative pool, and WMEB-weight corresponds to WMEB with the pool and no weighting. The weighting is extremely effective with an approximate 7% performance gain over MEB. The pool also noticeably improved performance, especially in the later iterations where it is needed most. These two components combine effectively together to significantly outperform MEB and BASILISK.

Our last evaluation is performed on the final test-set – the TREC Genomics full-text articles. We compare the performance of each algorithm on the TREC and MEDLINE collections (Table 11). WMEB performs consistently better than BASILISK and MEB, however each has a significant performance drop on TREC. This is due to the variation in language use. In abstracts, the content is more dense and precise, and thus contexts are likely to be less noisy. Full-text articles also contain less cohesive sections. In fact, ANTI with the largest performance drop for each algorithm (WMEB 95% MEDLINE, 30% TREC) extracted templates from the methods section, identifying companies that provide ANTI.

ALG	MEDLINE		TREC	
	Av(100)	Av(200)	Av(100)	Av(200)
BAS	78.4	74.7	63.0	57.9
MEB	76.3	70.1	55.2	49.6
WMEB	87.1	**89.5**	67.8	**66.0**

Table 11: Results comparing MEDLINE and TREC.

6 Conclusions

In this paper, we have proposed *Weighted Mutual Exclusion Bootstrapping* (WMEB), for efficient extraction of high precision lexicons, and the templates that identify them, from raw text. WMEB extracts the terms and templates of multiple categories simultaneously, based on the assumption of mutual exclusion. WMEB extends on MEB by incorporating more sophisticated scoring of terms and templates based on association strength and a cumulative template pool to keep templates active in the extraction process.

As a result, WMEB is significantly more effective at reducing semantic drift than MEB, which uses a simple weighting function, and BASILISK, which does not strictly enforce mutual exclusion between categories. We have evaluated these algorithms using a variety of semantic categories on three different raw text collections. We show that WMEB extracts more reliable large semantic lexicons than MEB and BASILISK (even with far fewer seeds). WMEB is more robust within the biomedical domain, which has an immediate need for these tools.

In the future, we plan to further investigate the mutual exclusion assumption, and whether it can be weakened to increase recall without suffering semantic drift, and how it interacts with the term and template scoring functions.

Our results demonstrate that WMEB can accurately and efficiently extract terms and templates in both general web text and domain-specific biomedical literature, and so will be useful in a wide range of NLP applications.

Acknowledgements

We would like to thank the anonymous reviewers and members of the LTRG at the University of Sydney, for their feedback. This work was supported by the CSIRO ICT Centre and ARC Discovery grant DP0665973.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical Report LDC2006T13, Linguistics Data Consortium.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *In Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 172–180.

Claire Grover, Michael Matthews, and Richard Tobin. 2006. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the Multi-dimensional Markup in Natural Language Processing (NLPXML)*, Trento, Italy.

Marti A. Hearst. 1992. Automatic aquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France, 23-28 July.

William Hersh, Aaron Cohen, L. Ruslen, and P. Roberts. 2007. TREC 2007 Genomics Track Overview. In *Proceedings of the 16th Text REtrieval Conference (TREC)*.

Rosie Jones, Andrew Mccallum, Kamal Nigam, and Ellen Riloff. 1999. Bootstrapping for text learning tasks. In *Proceedings of the IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*.

Christopher Manning and Hinrich Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT.

Edgar Meij and Sophia Katrenko. 2007. Bootstrapping language associated with biomedical entities. The AID group at TREC Genomics 2007. In *Proceedings of The 16th Text REtrieval Conference (TREC 2007)*.

Tara Murphy and James R. Curran. 2007. Experiments in mutual exclusion bootstrapping. In *Proceedings of the Australasian Language Technology Workshop (ALTW)*, pages 66–74.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the Conference on Computational Linguistics and the 46th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Sydney, Australia.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47, Philadelphia, USA.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pages 474–479, Orlando, FL, USA.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pages 117–124, Providence, RI.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 214–221, Philadelphia, USA.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical Report LDC2005T33, Linguistics Data Consortium.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational linguistics (COLING)*, pages 1135–1141, San Francisco.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 240–247.

David Yarowsky. 1995. Unsupervised word sense disambiguiation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Hong Yu and Eugene Agichtein. 2003. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19(1):i340–i349.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 106-114

Investigating Features for Classifying Noun Relations

Dominick Ng, David J. Kedziora, Terry T. W. Miu and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006 Australia
{dong7223,dked9370,tmiu9098,james}@it.usyd.edu.au

Abstract

Automated recognition of the semantic relationship between two nouns in a sentence is useful for a wide variety of tasks in NLP. Previous approaches have used kernel methods with semantic and lexical evidence for classification. We present a system based on a maximum entropy classifier which also considers both the grammatical dependencies in a sentence and significance information based on the Google Web 1T dataset.

We report results comparable with state of the art performance using limited data based on the SemEval 2007 shared task on nominal classification.

1 Introduction

Analysis of the semantics of natural language is an area of research undergoing renewed interest, driven by the many applications which can directly benefit from such information, including question answering and text summarization. In particular, recognition of the relationship between words in a sentence is useful for clarifying ambiguities in tools that attempt to interpret and respond to natural language. Recent developments in the semantic web – a vision where information is comprehensible to machines as well as people – have also accelerated the need for tools which can automatically analyse nominal relationships.

We approach the task of relation classification using the relation set and training data provided by the SemEval 2007 shared task on nominal classification (Girju et al., 2007). The problem as defined by the task description is to discover the underlying relationship between the concepts expressed by two nominals, excluding named entities. The relationship is informed by the context of an English sentence, e.g. it is clear that the relationship between door and car differs in the fragments the car door and the car scraped the door of the garage. Resolving the relation between nominals in cases where ambiguities exist is useful for generalisation in NLP systems.

We created a system based upon a maximum entropy (ME) classifier developed by Clark and Curran (2004). A separate binary classifier for each of the relations was trained over the corresponding training data, and the additional features used for each relation were selected by performing a seven-fold cross-validation over all combinations of features developed for this task. We report an overall accuracy of 71.9% macro-averaged over the seven relations and an overall F-measure of 70.7%, comparable with state of the art performance.

2 Background

The task of relation classification is complicated by the lack of consensus on relation sets and algorithms. Previous research has studied areas as diverse as noun compound classification in the medical domain (Rosario and Hearst, 2001), gene relations (Stephens et al., 2001), verb-verb semantic relations (Chklovski and Pantel, 2004), and noun-modifier relations (Nastase and Szpakowicz, 2003). Many independent class hierarchies have been developed to suit each application domain, and it is difficult to transfer one

Relation	Training Example
Cause-Effect	[e_1 Famine] following [e_2 drought] has hit the West African savannahs, where there have been other bad droughts.
Instrument-Agency	The [e_1 judge] hesitates, [e_2 gavel] poised.
Product-Producer	The [e_1 artist] made the [e_2 picture] when he was in fourth grade.
Origin-Entity	It's unfortunate you didn't try a [e_1 potato] [e_2 vodka].
Theme-Tool	The [e_1 submission] [e_2 deadline] is February, 2, 2007.
Part-Whole	Typically, an unglazed [e_1 clay] [e_2 pot] is submerged for 15 to 30 minutes to absorb water.
Content-Container	The [e_1 kitchen] holds patient [e_2 drinks] and snacks.

Table 1: Examples of the SemEval relations.

of these hierarchies to another domain. The organisers of the SemEval task defined the classification problem in terms of seven semantic relations commonly mentioned by researchers, and a list of these along with some training examples is provided in Table 1. An annotated dataset of 140 training examples and at least 70 test sentences was created for each relation by searching the web using wild-card search patterns satisfying the constraints of each relation, e.g. * holds * for the Content-Container relation. This method was used in order to provide near miss negative examples (Girju et al., 2007).

Fifteen systems split into four categories were submitted for the SemEval 2007 workshop. Almost all of the systems utilised extended feature sets that built upon the data provided by the task; most systems also implemented some form of statistical or kernel approach to develop binary classifiers for the relations (see Bedmar et al. (2007), Hendrickx et al. (2007), and Nulty (2007) for some previous approaches to the classification task explored in this paper). The best performing systems achieved F-measures in the range of 71.5% – 72.4% by utilising the provided WordNet sense keys and adding more training examples to those supplied by the task; however, the majority of systems did not augment the provided data like this and reported F-measures and accuracies below 67.2% (Girju et al., 2007).

Each training example in the annotated dataset consists of a sentence, two nominals whose relationship is to be evaluated, WordNet 3.0 sense keys for each of the nominals, the wild-card query used to obtain the example, and comments on the choices made during the creation of the example. The evaluation drew distinction between systems that did and did not use the supplied WordNet and wild-card query information.

3 Maximum Entropy Modelling

The entropy of a classifier is a measure of how predictable that classifier's decisions are. The lower the entropy, the more biased a classifier is, i.e. a relation classifier has zero entropy if it always assigns the same relation to any input. The theory underpinning ME modelling is that the distribution chosen to fit the specified constraints will eliminate biases by being as uniform as possible. Such models are useful in NLP applications because they can effectively incorporate diverse and overlapping features whilst also addressing statistical dependencies.

We used the ME implementation described in Clark and Curran (2004). The ME models used have the following form:

$$p(y|x,\lambda) = \frac{1}{Z(x|\lambda)}\exp\left(\sum_{k=1}^{n}\lambda_k f_k(x,y)\right)$$

where $Z(x|\lambda)$ is the normalisation function and the f_k are features with associated weights λ_k. The system uses Gaussian smoothing on the parameters of the model. The features are binary-valued functions which pair a relation y with various observations x from the context provided, e.g.

$$f_j(x,y) = \begin{cases} 1 & \text{if } word(x) = damage \text{ \&} \\ & y = \text{Cause-Effect-True} \\ 0 & \text{otherwise} \end{cases}$$

4 Features and Methodology

We focused our efforts on finding features which aggressively generalise the initial material over as broad a search space as possible. We investigated lexical, semantic, and statistical features sourced from a number of corpora as well as morpho-syntactic features from a grammatical parse of each sentence. Features were evaluated using a seven-fold cross-validation performed over the training data for each relation over every possible combination of features – a process made possible by the small size of the corpora and the relatively small number of features experimented with. The speed of the training process for the ME implementation was also a factor in enabling the exhaustive search.

The features which resulted in the best performance for each relation in the cross-validation were then used to train seven binary classifiers for the final run over the supplied test data.

4.1 Preprocessing

Prior to feature generation our system extracted from the supplied data the sentences and marked nominals (termed as e_1 and e_2). While WordNet was used internally as features by our system, we did not use the specific sense keys provided by the data to query WordNet – we relied upon a more general word lookup that extracted all of the possible senses for the nominals. We also did not make use of the provided query, based on its ineffectiveness in previous studies (Girju et al., 2007).

Close examination of the training data also revealed some negative training examples that were identified in comments as belonging to a different relation set. These examples were collected and added to the appropriate training file to further extend the original dataset. However, no new examples were created: only examples which had already been identified as belonging to a particular relation were added in this process.

4.2 Lexical Features

Lexical features are useful for capturing contextual information about the training example, and they are the most obvious features to incorporate. However, due to the limited amount of training data available for this task, lexical features encounter sparseness problems as there are few relevant collisions between words. We utilised the following lexical features:

- **sen**: The words of the sentence itself. This was used as the baseline for feature testing;
- **red**: A reduced version of the sentence with all words of length 2 or less removed;
- **heads**: The head words of the nominals in question, e.g. for [e_1 tumor shrinkage] after [e_2 radiation therapy] the relation actually holds between shrinkage and therapy. This feature is very specific, but allows for nominals which are commonly linked to certain relations to be identified;
- **dir**: The required direction of the relation (i.e. from e_1 to e_2 or vice versa) that is encoded in the data – useful as some relations are more likely to exist in a particular direction, e.g. the Part-Whole relation is most commonly found encoded in the direction [e_1 Part]-[e_2 Whole] (Beamer et al., 2007).

4.3 WordNet Features

WordNet (Fellbaum, 1998) is the most heavily used database of lexical semantics in NLP. Created at Princeton University, WordNet is based around groups of synonyms (synsets) and encodes a vast array of semantic properties and relationships between these synsets. The coverage of WordNet means that it is very useful for generalising features over a small corpus of data, and many previous approaches to classification tasks have utilised WordNet in some way – including most of the systems from the SemEval proceedings (Girju et al., 2007). However, unlike most of these systems, we did not use the supplied WordNet sense keys as we believe that it is unrealistic to have such precise data in real-world applications. As a consequence, all of our WordNet features were extracted using the indicated nominals as query points.

- **syn**: Synonyms of the nominals. We extracted from WordNet all synonyms in all senses for each of the marked nouns;
- **hyp1, hyp2**: Hypernyms of the nominals, i.e. more general concepts which encompass the nominals. These features allow us to

broaden the coverage given by the nominals over less specific entities. We exhaustively mined all hypernyms of the marked nouns to a height of two levels, and encoded the two levels as separate features;

- **lex**: Lexical file numbers, which correspond to a number of abstract semantic classes in WordNet, including noun.artifact, noun.event, and noun.process. This allows for nominal relations which do not make sense to be identified, e.g. a noun.process should not be able to contain a noun.event, but the process may cause the event (Bedmar et al., 2007);

- **cont**: Container - a binary feature indicating whether the marked nouns are hyponyms (more specific concepts) of the container synset. This feature was included mainly for the benefit of the Content-Container relation; however, we hypothesised that their inclusion may also assist in classifying other relations; e.g. the 'effect' in Cause-Effect should not be a physical entity.

4.4 Grammatical Relations Features

Syntactic features representing the *path* between nominals are a useful complement for semantic and lexical features because they account for the way in which words are commonly used in text. Semantic relationships can often be associated with certain patterns of words, e.g. the pattern e_1 is inside e_2 is a strong indicator for the Content-Container relation for many general combinations of e_1 and e_2. However, these patterns can be expressed in many different ways - inside e_2 e_1 is or inside e_2 is e_1 are other ways of expressing a Content-Container relationship – and while the words are essentially the same between the examples the changed ordering creates difficulties in designing good features. This problem can be alleviated by considering syntactic dependencies in a sentence rather than a naive concatenation of words (Nicolae et al., 2007).

Grammatical relations (GRs) represent the syntactic dependencies that hold between a head and a dependent in text. Initially proposed by Carroll et al. (1998) as a framework-independent metric for parsing accuracy, GRs are arranged in a hierarchy that allows for varying levels of exactness in parsing: a general *dependent* relation can be assigned to indicate that there is some doubt over the precise dependency that holds between two words. We postulated that a simple graph constructed from the dependencies (whereby words of the text are nodes and undirected edges are added between nodes if there is some grammatical relation that links them) could be used to find a path between the two nominals in each sentence. This path would compare favourably to a naive concatenation of the words between the nominals as it considers the actual dependencies in the sentence rather than just the positions of the words, although in many cases at least one of the words between the marked nominals in the sentence will be represented in the dependency path. Table 2 gives a list of GRs used in this process.

GRs	Description
conj	coordinator
aux	auxiliary
det	determiner
ncmod	non-clausal modifier
xmod	unsaturated predicative modifier
cmod	saturated clausal modifier
pmod	PP modifier with a PP complement
ncsubj	non-clausal subject
xsubj	unsaturated predicative subject
csubj	saturated clausal subject
dobj	direct object
obj2	second object
iobj	indirect object
pcomp	PP which is a PP complement
xcomp	unsaturated VP complement
ccomp	saturated clausal complement
ta	textual adjunct delimited by punctuation

Table 2: A list of GRs

```
(det man_1 A_0)
(ncmod _ does_2 not_3)
(aux talk_4 does_2)
(ncsubj talk_4 man_1 _)
(det woman_7 every_6)
(ncsubj walks_8 woman_7 _)
(conj or_5 walks_8)
(conj or_5 does_2)
```

Figure 1: GRs output from the C&C parser

To extract the grammatical relations from the provided data we parsed each training and test example with the C&C parser developed by Clark and Curran (2007). Figure 1 gives an example of the GRs for the sentence A man does not talk or every woman walks. A dependency graph was generated from this output and the shortest path between the nominals found. In the example in Figure 1, the path between man and woman is talk_does_or_walks.

Features were extracted from this path output in two formats: a generalised version (labelled with a 'g' prefix), whereby the two nominals in question were replaced whenever they appeared with the marker tags e_1 and e_2, and the actual version, where this extra generalisation step was not applied. We reasoned that the generalised output would be more useful as a classification feature as it removed the stipulation on the start and end of the path; however, we also felt that keeping the identity of the nominals would aid in classifying words often paired with prepositions that suggest some form of spatial or logical relationship, e.g. the fragment after_hurricanes suggests some form of Cause-Effect relationship from the temporal indicator 'after'. Our path features included:

- **path, gpath**: The path itself in a concatenated format, e.g. damage_comes_after_hurricanes or e_1_comes_after_e_2. These patterns were postulated to have some correlation with each relation;
- **strip, gstrip**: The path with a length filter of 2 applied;
- **slice, gslice**: The nominals with their immediate neighbour from the path, e.g. damage_comes, after_hurricanes or e_1_comes, after_e_2;
- **pair, gpair**: The bigrams in the path, e.g. e_1_comes, comes_after, after_e_2;
- **ptag, gptag**: The underscore-concatenated POS tags of the path words.

4.5 Web 1T Significance Features

Web 1T (Brants and Franz, 2006) is a Google-released corpus containing English word ngrams and their observed frequency counts in a body of approximately 1 trillion word tokens of text from publicly accessible web pages. Web 1T counts the occurrences of unigrams, 2-, 3-, 4-, and 5-grams in the 1 trillion word tokens, discarding unigrams appearing less than 200 times in the tokens (1 in 5 billion) and n-grams appearing less than 40 times (1 in 25 billion) in the tokens. This resource captures many lexical patterns used in common English, though there are some inconsistencies due to the permissive nature of the web: some commonly misspelt words are included and some text in languages other than English are also present.

O_{11}: *freq count of word and bound together*
O_{12}: *freq count of bound without word*
O_{21}: *freq count of word without bound*
O_{22}: *freq count of neither bound or word*
N: *total number of tokens*

$$\chi^2 = \frac{N(O_{11}O_{22}-O_{12}O_{21})^2}{(O_{11}+O_{12})\times(O_{11}+O_{21})\times(O_{12}+O_{22})\times(O_{21}+O_{22})}$$

Figure 2: The notation and formula used for the significance testing.

The idea of searching a large corpus for specific lexical patterns to indicate semantic relations of interest was first described by Hearst (1992). As previously mentioned, we postulated that certain patterns of words would associate with certain relations, but a naive concatenation of words located between the nominals would be unhelpful with such a small data set. This problem can be avoided by examining the frequencies of lexical patterns within a much larger dataset such as Web 1T, where the problem of data sparseness is offset by the size of the corpus. This pattern information would complement the semantic and syntactic information already used by incorporating evidence regarding word use in real-world text.

We chose to conduct statistical significance tests with the intention of observing if the presence of particular words between the nominals is meaningful, irrespective of whether or not they are in the sentences themselves. This allows us to collate all the words found to be significant when placed between the nominals and use them as ngram features. Our methodology is justified by the observation that patterns correlated with relations are likely to contain the same words regardless of the bounds, i.e. the pattern e_1 is inside

Baseline	F± std
sen	61.3±9.29
Feature	Mean Improv± std
red	+0.3±1.67
heads	+0.7±2.47
dir	+0.3±2.10
syn	+1.2±3.29
hyp1	+1.3±4.25
hyp2	+5.6±4.39
lex	+6.0±6.75
cont	+1.5±2.52
path	+0.2±1.00
gpath	+0.5±3.17
strip	+0.2±1.00
gstrip	-0.5±3.47
slice	+1.5±3.03
gslice	+0.5±2.88
pair	+0.2±1.93
gpair	+1.4±2.82
ptag	+0.5±1.54
gptag	-0.6±2.79
ngram	+1.8±2.94

Table 3: The average improvement in F-measure (using the words of the sentence as a baseline) for each feature macro-averaged over all 7 relations

e_2 is a strong indicator for the Content-Container relation for general combinations of e_1 and e_2.

The significance test was conducted as in Manning and Schütze (2000). The Web 1T data was searched for any 3-, 4-, and 5-grams that had the same bounds as the nominals of the sentence in question, i.e. patterns which match e_1 ... e_2. Then, for every intermediate word in the pattern, a χ-squared value was calculated to measure the significance of the word in relation to the bounds. This process was repeated for each training example, and Figure 2 gives the equations used for this test. We conducted some brief experiments to find the range of χ-squared values returned by this test; based on these we chose the χ-squared value of 10 to indicate significance to the training example being analysed, and selected all words with a χ-squared value above this level to add as ngram features.

5 Preliminary Feature Testing

As an initial step, we used the sentence words of each example as a baseline to test the individual performance of each feature in a seven-fold cross-validation over the training examples for each relation. We did this to compare the discrete improvement over the baseline that each feature offered and to allow a comparison as to how combining the features improves performance.

Relation	Best	Improvement
Cause-Effect	lex	+8.97
Instrument-Agency	hyp2	+7.31
Product-Producer	hyp1	+5.68
Origin-Entity	lex	+12.96
Theme-Tool	lex	+16.45
Part-Whole	hyp1	+2.95
Content-Container	hyp2	+6.84

Table 4: The best performing single features (using the words of the sentence as a baseline) and their mean improvement in F-measure for each relation

Table 3 shows that most features offer small gains on average over the baseline sentence, but also exhibit varying degrees of performance over the relations as seen in the relatively large standard deviations. In particular, lexical file numbers and second-level hypernyms have the largest mean improvement in F-measure, but also the largest standard deviations – indicating a widespread distribution of positive and negative contributions. Table 4 shows that these two features improve the baseline F-measure of five of the relations, implying from the large standard deviations that they severely worsen the performance of the remaining two. This behaviour is explained by noting the wide generalisation that these features add, creating the most collisions between training and test data and hence affecting the decisions of the classifier the most.

6 Results and Discussion

The features chosen to train the classifiers for the final system along with performance in the cross-validation are given in Table 5. All the relations performed best with a combination of lexical, semantic, and syntactic features, and three relations also used the statistical significance data obtained from Web 1T. The relatively even spread of feature types across relations implies that the classifier performs best when presented with a wide range of evidence that it can then combine into a model. However, the largest number of fea-

Relation	Features selected	F± std	Acc± std
Cause-Effect	dir, heads, cont, lex, pair, g/slice, g/strip, ngram	77.9± 7.06	73.6±9.00
Instrument-Agency	heads, cont, hyp2, lex, g/pair, gptag, gpath, g/slice, gstrip	78.2± 7.59	77.1±9.94
Product-Producer	heads, red, cont, hyp1, hyp2, gpath, pair, slice, strip	80.6± 3.67	72.1±3.93
Origin-Entity	dir, sen, hyp2, lex, gstrip	62.3±12.92	70.7±8.38
Theme-Tool	heads, lex, g/pair, gslice	71.2± 8.43	77.9±4.88
Part-Whole	dir, red, hyp1, syn, g/pair, slice, ngram	81.6± 8.81	77.1±8.09
Content-Container	heads, red, cont, hyp2, lex, pair, slice, ngram	71.1±12.22	72.9±6.36
Average	–	74.7± 6.88	74.5±2.85

Table 5: The best performing features with F-measure and accuracy percentages from the cross-validation

tures used was 11 – considerably less than the 20 tested during cross-validation – and this supports the general conclusion that using too many features in a maximum entropy approach with a small amount of training data adversely affects classifying performance.

The most commonly selected features were the Grammatical Relations bigram features (pair and g / slice). These features were used in all but one of the classifiers, indicating that bigram information provided very useful evidence for relation classification. Given that most path bigrams involve the nominals with a preposition that indicates a temporal or spatial relationship, we infer that the syntactic dependency between nominals and prepositions is an important feature for semantic relation classification. Other commonly selected features were the head words of the sentence and their lexical file numbers – these were present together in the Cause-Effect, Instrument-Agency, Theme-Tool, and Content-Container classifiers. This correlation is expected given that these relations usually exist between nominals that generally correspond with the semantic classes from WordNet.

Table 5 shows that some relations were more challenging to classify than others. Origin-Entity in particular exhibited the worst performance, with a standard deviation of 12.92 around an average F-measure of 62.3% under seven-fold cross-validation. This poor performance was expected given that most attempts from the SemEval proceedings rated Origin-Entity as equal hardest to classify along with Theme-Tool (Girju et al., 2007). On the other hand, our cross-validation yielded good performance for Theme-Tool, with an F-measure of 71.2% – potentially showing that maximum entropy methods are more effective at handling difficult relations than kernel approaches to the problem. Also notable are the strong performances of the Part-Whole and Product-Producer classifiers, with F-measures above 80% and accuracies above 72%. The other relations also performed well, with no other classifier exhibiting an F-measure or accuracy score below 70%.

Relation	P	R	F	Acc
Cause-Effect	78.0	69.6	73.6	71.3
Instrument-Agency	84.2	72.7	78.1	76.9
Product-Producer	87.1	70.1	77.7	66.7
Origin-Entity	50.0	75.0	60.0	70.4
Theme-Tool	62.1	72.0	66.7	74.6
Part-Whole	80.8	51.2	62.7	65.3
Content-Container	65.8	89.3	75.8	78.4
Average	**72.6**	**71.4**	**70.7**	**71.9**

Table 6: Final percentage precision, recall, F-measure, and accuracy results over the test data using the features listed in Table 5

Table 6 gives the final classifying results over the supplied test data using all the training examples and features selected in the cross-validation step as training material. We established a new benchmark for classifying the Instrument-Agency relation: our F-measure of 78.1% exceeds the best result of 77.9% for the relation from the SemEval proceedings (Girju et al., 2007). However, as a general rule, system performance was weaker over the test data than during the cross-validation step, providing some evidence of overfitting to the training data. This was particularly demonstrated in the markedly poor performance of the Part-Whole classifier – from the cross-validation F-measure dropped by 18.9% to 62.7% and accuracy fell 12.4% from 77.1% to 65.3%. It should be noted however that our system performed better than most others in classifying the difficult relations as ranked in the SemEval task (Girju et al., 2007).

We recorded a final F-measure of 70.7% and accuracy of 71.9% macroaveraged over the seven relations, an improvement of 5.9% in both F-measure and accuracy over the best performing system using the same data (no WordNet sense keys or query) from SemEval 2007. Our system performed within an F-measure of 1.7% and accuracy of 4.4% of the top system from SemEval 2007, which incorporated a large number of extra training examples and WordNet sense keys (Beamer et al., 2007). Our results are comparable with more recent approaches to the same classification task, utilising pattern clusters (F-measure 70.6%, accuracy 70.1%, in Davidov and Rappoport (2008)) and distributional kernels (F-measure 68.8%, accuracy 71.4%, in Séaghdha and Copestake (2008)).

Overall these results show that a maximum entropy approach with a range of informative features is a feasible and effective method of classifying nominal relations when presented with limited data.

7 Conclusion

We have created a system built around a maximum entropy classifier that achieves results comparable with state-of-the-art with limited training data. We have also demonstrated that syntactic dependencies and frequency-based statistical features taken from large corpora provide useful evidence for classification, especially when combined with lexical and semantic information.

We have also shown that a maximum entropy approach using informative features performs strongly in the task of relation classification, and that exact WordNet sense keys are not necessary for good performance. This is important since it is impractical in large scale classifying tasks to provide this annotation.

The corpora is extremely small, and it should be noted that the choice to select the dataset using a limited number of queries artificially limits the scope of this task. We feel that an effort to annotate a large amount of randomly selected text with several hundred positive examples would greatly benefit further research into relation classification and validate the results presented in this paper.

Future improvements to the system could include incorporating more external resources (e.g. VerbNet), introducing Word-Sense Disambiguation as a replacement for WordNet sense keys, or by incorporating more relation-specific features, such as meronym (Has-Part) information from WordNet for the Part-Whole and Content-Container relations. More sophisticated analysis of the Web 1T data could also be undertaken, such as a generalised attempt to identify patterns underpinning semantic relationships, rather than just those corresponding to the provided sentences.

We achieved a final overall F-measure of 70.7% and accuracy of 71.9%, establishing a new benchmark for performance over the SemEval data without sense keys. Our system is also competitive with approaches that use sense keys, and so we expect that it will provide useful semantic information for classification and retrieval problems in the future.

Acknowledgments

The authors would like to thank the three anonymous reviewers, whose comments greatly improved the quality of this paper. Dominick Ng was supported by a University of Sydney Merit Scholarship; Terry Miu was supported by a University of Sydney Outstanding Achievement Scholarship and a University of Sydney International Merit Scholarship.

References

Brandon Beamer, Suma Bhat, Brant Chee, Andrew Fister, Alla Rozovskaya, and Roxana Girju. 2007. UIUC: A knowledge-rich approach to identifying semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 386–389, Prague, Czech Republic.

Isabel Segura Bedmar, Doaa Samy, and Jose L. Martinez. 2007. UC3M: Classification of semantic relations between nominals using sequential minimal optimization. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 382–385, Prague, Czech Republic.

Thorsten Brants and Alex Franz. 2006. Web 1T

5-gram version 1. Linguistic Data Consortium, Philadelphia. LDC2006T13.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings, First International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods on Natural Language Processing (EMNLP-2004)*, pages 33–40, Barcelona, Spain.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 104–111, Barcelona, Spain.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Dmitry Davidov and Ari Rappoport. 2008. Classification of semantic relationships between nominals using pattern clusters. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 227–235, Ohio, USA.

Christiane Fellbaum, editor. 1998. *WordNet - An Electronic Lexical Database*. MIT Press.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18, Prague, Czech Republic.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-1992)*, pages 539–545, Nantes, France.

Iris Hendrickx, Roser Morante, Caroline Sporleder, and Antal van den Bosch. 2007. ILK: Machine learning of semantic relations with shallow features and almost no data. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 187–190, Prague, Czech Republic.

Chris Manning and Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing*. MIT Press, Massachusetts, United States.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the 5th International Workshop on Computational Semantics*, pages 285–301, Tilburg, The Netherlands.

Cristina Nicolae, Gabriel Nicolae, and Sanda Harabagiu. 2007. UTD-HLT-CG: Semantic architecture for metonymy resolution and classification of nominal relations. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 454–459, Prague, Czech Republic.

Paul Nulty. 2007. UCD-PN: Classification of semantic relations between nominals using wordnet and web counts. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 374–377, Prague, Czech Republic.

Barbara Rosario and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-2001)*, pages 82–90, Pennsylvania, United States.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pages 649–655, Manchester, UK.

Matthew Stephens, Mathew J. Palakal, Snehasis Mukhopadhyay, Rajeev R. Raje, and Javed Mostafa. 2001. Detecting gene relations from MEDLINE abstracts. In *Proceedings of the Sixth Annual Pacific Symposium on Biocomputing*, pages 483–496, Hawaii, United States.

Learning Count Classifier Preferences of Malay Nouns

Jeremy Nicholson and **Timothy Baldwin**
NICTA Victoria Research Laboratories
University of Melbourne, VIC 3010, Australia
{jeremymn,tim}@csse.unimelb.edu.au

Abstract

We develop a data set of Malay lexemes labelled with count classifiers, that are attested in raw or lemmatised corpora. A maximum entropy classifier based on simple, language-inspecific features generated from context tokens achieves about 50% F-score, or about 65% precision when a suite of binary classifiers is built to aid multi-class prediction of headword nouns. Surprisingly, numeric features are not observed to aid classification. This system represents a useful step for semi-supervised lexicography across a range of languages.

1 Introduction

This work examines deep lexical acquisition (DLA) of count classifiers in Malay.[1] DLA is the process of (semi-)automatically learning linguistic structures for use in linguistically-rich language resources such as precision grammars or wordnets (Baldwin, 2007). Malay is a significant target for DLA in that relatively few NLP resources have been developed for it.[2]

In many languages — notably many South-East and East Asian languages — numbers cannot generally modify nouns. Instead, they modify count classifiers[3] which tend to occur as a genitive modifier or in apposition to the noun. In (1) below, *biji* is an example of a Malay count classifier (CL), premodified by the numeral *tiga* "three"; similarly for the Japanese in (2):[4]

(1)	*tiga*	*biji*	*pisang*
	three	CL	banana
	"three bananas"		

(2)	*saN*	*boN*	*no*	*banana*
	three	CL	GEN	banana
	"three bananas"			

The closest English analogue to count classifiers is measure nouns (Flickinger and Bond, 2003), like *loaf* in *3 loaves of bread* (compare *3 breads*, which has markedly different semantics). Measure nouns cannot be used to count countable nouns in English, however (e.g. ∗*3 loaves of roll*).

Syntactically, count classifiers are nouns which occur with a (usually numeric) specifier, and are used to quantify uncountable nouns. In languages such as Malay, most nouns are uncountable and numeral–classifier combinations must be used to enumerate instances of a given noun (c.f. ∗*tiga pisang*; this is not true of the nouns that are classifiers themselves, like *orang* "person" or *buah* "fruit", e.g. *tiga orang* "three people"). Semantically, count classifiers select for objects of particular semantics, commonly determined by a conceptual class (e.g. HUMAN or BIRD) or the relative dimensions of the object (e.g. LONG-AND-THIN or FLAT).

[1]The language is called by a number of names, including bahasa Malayu, bahasa Malaysia, the Malaysian language, and Standard Malay. We simply use *Malay*, to agree with the Ethnologue (Gordon, 2005).

[2]Although some research has been done on bahasa Indonesia (henceforth *Indonesian*), another Malayan language with which Malay is (somewhat) mutually intelligible.

[3]This paper focuses specifically on sortal classifiers. See Paik and Bond (2001) for discussion of other types of count classifiers, namely event, mensural, group and taxonomic.

[4]In the Japanese example, the genitive (GEN) linker is required to connect the numeral–classifier combination with the noun in a single noun phrase.

In the case of Malay, e.g., *biji* is the count classifier for FRUIT, while *orang* is the count classifier for HUMAN:

(3) *empat orang raja*
four CL king
"four kings"

(4) *#empat biji raja*
four CL king
"four kings" (intended)

There are over 50 such count classifiers in Malay, each with differing semantic restrictions.

Count classifiers are highly significant in languages such as Malay when generating text, because a noun specified with an incorrect classifier can lead to unintended or infelicitous readings (c.f. *#empat biji raja* above), or highly marked language. Other languages, like Thai, require classifiers not only for specification but also stative verb modification. Additionally, the choice of classifier can help disambiguate noun sense, either for unknown nouns or for known ambiguous nouns. In order to generate correctly or use count classifiers as a source of disambiguation, however, we require a large-coverage noun lexicon with information on classifier compatibility. The objective of this paper is to automatically generate such a lexicon for Malay. As far as we are aware, this is the first such attempt for Malay.

The proposed approach to learning the classifier preferences for Malay nouns is to represent nouns via their distributional context, i.e. the tokens they commonly co-occur with in corpus data within a fixed context window. We model distributional context in a standard supervised learning framework, generalising from training examples to classify novel nouns. In this, we experiment with different distributional representations (words vs. lemmas, and different n-gram orders), and compare our proposed system to a system based on direct analysis of noun co-occurrence with different numeral–classifier combinations.

This research forms part of a broader project on the general applicability of DLA across a range of word learning tasks. As such, we propose a model that is as language-inspecific as possible, making only the bare minimum assumptions about Malay such as the fact that a modifier tends to occur within a few tokens of its head. As a result, we expect that the observed results will scale to other languages with count classifier systems such as Japanese, Chinese, or Thai. The major difference in porting the method to other languages would be access to various lexical resources — for example, lexical acquisition would be quite difficult for these three languages without a tokeniser — where available parsers or ontologies might capture syntactic or semantic similarities much more easily than surface cues.

As far as we are aware, this paper presents the first ever results for count classifier-based classification of Malay nouns.

2 Background

2.1 NLP for Malay

As the basis of our evaluation, we use the KAMI Malay–English translation dictionary (Quah et al., 2001). Although primarily a translation dictionary, with translations of Malay headwords into both English and Chinese, KAMI contains syntactic information and semantics in terms of a large ontology. Of the total of around 90K lexical entries in KAMI, we make use of around 19K nominal lexical entries which are annotated for headword, lemma and POS tag when specified, and count classifier.

As a corpus of Malay text, we use the 1.2M-token web document collection described in Baldwin and Awab (2006).[5] The corpus has been sentence- and word-tokenised, and additionally lemmatised based on a hand-crafted set of inflectional and derivational morphology rules for Malay, developed using KAMI and an electronic version of a standard Malay dictionary (Taharin, 1996).

While little NLP has been performed for Malay, Indonesian has seen some interesting work in the past few years. Adriani et al. (2007) examined stemming Indonesian, somewhat overlapping with Baldwin and Awab above, evaluating on the information retrieval testbed from Asian et al. (2004). Recently, a probabilistic parser of Indonesian has been developed, as discussed in Gusmita and Manu-

[5]An alternative corpus of Malay would have been the Dewan Bahasa & Pustaka Corpus, with about 114M word tokens. As it is not readily accessible, however, we were unable to use it in this research.

rung (2008), and used for information extraction and question answering (Larasati and Manurung, 2007).

2.2 Count Classifiers

Shirai et al. (2008) discuss the theory and difficulties associated with automatically developing a broad taxonomy of count classifiers suitable for three languages: Chinese, Japanese, and Thai. They find that relatively high agreement between Chinese and Japanese, but Thai remains resistant to a universal hierarchy of count classes.

Otherwise, work on count classifiers has mostly focussed on a single language at a time, primarily Japanese. Bond and Paik (1997) consider the lexical implications of the typology of kind and shape classifiers, and propose a hierarchy for Japanese classifiers that they extend to Korean classifiers. Bond and Paik (2000) examine using the semantic class of a Japanese noun from an ontology to predict its count classifier, and Paik and Bond (2001) extend this strategy to include both Japanese and Korean. Finally, Sornlertlamvanich et al. (1994) propose their own typology for classifiers within Thai, and an automatic method to predict these using corpus evidence.

As for comparable structures in English, Flickinger and Bond (2003) analyse measure nouns within a Head-Driven Phrase Structure Grammar framework. Countability, which is a salient feature of nouns in English, undergoes many of the same structural syntax and semantic preferences as count classifiers. Baldwin and Bond (2003) motivate and examine a variety of surface cues in English that can be used to predict typewise countability of a given noun. Taking a mapped cross-lingual ontology, van der Beek and Baldwin (2004) use the relatedness of Dutch and English to cross-lingually predict countability, and observe comparable performance to monolingual prediction.

3 Methodology

3.1 Data Set

First, we constructed a data set based on the resources available. As we chose to approach the task in a supervised learning framework, we required a set of labelled exemplars in order to train and test our classifiers.

To obtain this, we first extracted the approximately 19K noun entries from KAMI (Quah et al., 2001) for which at least one count classifier was attested. For each of these, we recorded a wordform and a lemma. The wordform was extracted directly from the lexical entry, and included both simplex words and multiword expressions. The lemma also came directly from the lexical entry where a lemma was provided, and in instances where no lemma was found in KAMI, we used a case-folded version of the headword as the lemma. We looked up each noun in the corpus based on the wordform in the original text data, and also the lemma in the lemmatised version of the corpus.

The gold-standard data set was generated from those lexical entries for which we were able to find at least one instance of the wordform in the raw corpus or at least one instance of the lemma in the lemmatised corpus. This resulted in a total of 3935 unique nouns. The total number of number classifiers attested across all the nouns was 51.

We also generated three reduced data sets, by: (1) excluding proper nouns, based either on the POS tag in KAMI or in the case that no POS was found, the capitalisation of the headword (3767 unique nouns); (2) excluding multiword expressions (MWEs: Sag et al. (2002)), as defined by the occurrence of whitespace in the headword (e.g. *abang ipar* "brother-in-law"; 2938 unique nouns); and (3) excluding both proper nouns and MWEs (2795 unique nouns). The underlying assumption was that proper nouns and MWEs tend to have obfuscatory syntax or semantics which could make the task more difficult; it would seem that proper nouns and MWEs should be handled using a dedicated mechanism (e.g. choosing the head of a compound noun from which to select a classifier).

3.2 Features

For each exemplar within the gold-standard data set, we developed a feature vector based on instances from the corpus. To limit the language specificity of the feature set, we built features by considering a context window of four tokens to the left and right for each corpus instance. The rationale for this is that while numerous languages permit free global word order, local word order (say, of an NP) tends to be more rigid. Windows of sizes less than four tended to reduce performance slightly; larger windows were not observed to significantly change performance.

For each instance of the wordform (i.e. the headword or headwords of a lexical entry from KAMI) in the raw corpus, we generated a feature vector of each of the (up to) four preceding and (up to) four following wordform tokens within the boundaries of the sentence the wordform occurred in. We additionally index each context wordform by its relative position to the target word. The same was done with the lemma of the target word, using the lemmatised corpus instead of the raw corpus. Consequently, there were numerous mismatches where a context window appeared in the wordform features and not the lemma features (e.g. if a morphologically-derived wordform was not listed in KAMI with its lemma), or *vice versa*. The combined feature vector for a given noun instance was formed by concatenating all of the context window features, as well as the target word wordform and lemma.

One possible extension that we chose not to follow on the grounds of language specificity is the use of morphological features. Malay has a number of prefixes, suffixes, and circumfixes which could provide information for selecting a classifiers. Specifically, the agentive prefix *me-* could be indicative of the *orang* class.

3.3 Classifiers

To build our classifiers, we used a maximum-entropy learner.[6] It is not uncommon for a noun to be listed with multiple classifiers in KAMI, such that we need a classifier architecture which supports multi-class classification. We experimented with two modes of classification: a monolithic classifier, and a suite of binary classifiers.

For the monolithic classifier, we adopt the equivalent of a first-sense classifier in word sense disambiguation terms, that is we label each training instance with only the first count classifier listed in KAMI. The underlying assumption here is that the first-listed count classifier is the default for that noun, and thus likely to have the best fit with the distributional model of that noun. As such, out of all the possible count classifiers, we expect to have the best chance of correctly predicting the first-listed classifier, and also expect it to provide the best-quality training data. When we restrict the set of annotations to only the first-listed classifiers, we reduce the set of possible classes to 42, out of which 10 are singletons and therefore never going to be correctly classified. In evaluating the monolithic classifier, we calculate precision and recall across the full set of actual count classifiers listed in KAMI (i.e. not just the first-listed count classifier).

For the suite of binary classifiers, we construct a single classifier for each of the 51 classes (of which 16 are singleton classes, and hence disregarded). For a given classifier, we categorised each exemplar according to whether the particular class is an acceptable count classifier for the headword according to KAMI. In testing, the classifier posits an affirmative class assignment when the posterior probability of the exemplar being a member of that class is greater than that of it not being a member. This mode of classification allows multi-classification more easily than that of the monolithic classifier, where multi-classes are too sparse to be meaningful. Evaluation of the suite is in terms of precision and recall averaged across the entire set of classifiers.

When selecting the training and test splits, we use 10-fold stratified cross-validation. Briefly, we separate the entire data set into 10 distinct partitions which each have generally the same class distribution as the entire data set. Each of the 10 partitions is used as a test set, with the other 9 as training partitions, and the results are micro-averaged across the 10 test sets. This evaluation strategy is used for both the monolithic classifier and the binary suite. 10-fold stratified cross-validation has been variously shown to tend to have less bias and variance than other hold-out methods, thereby giving a better indication of performance on unseen data.

4 Results

4.1 Comparison of Feature Sets

First, we examined the use of wordform features, lemma features and the combination of the two. We contrast the precision (P), recall (R), and F-score $F_{\beta=1} = \frac{2PR}{P+R}$ for the monolithic classifier (Mono) and the suite of binary classifiers (Suite) across the

[6] We used the OpenNLP implementation available at `http://www.sourceforge.net/projects/maxent/`.

				Mono			Suite		
Token	LEs	n-gram	Numb	P	R	F	P	R	F
Baseline	All			43.8	41.5	42.6	43.8	41.5	42.6
Baseline	−PN −MWE			42.4	39.3	40.8	42.4	39.3	40.8
W	All	1	No	48.3	45.2	46.7	76.4	33.7	46.8
L	All	1	No	49.7	47.0	48.3	61.0	38.3	47.1
W+L	All	1	No	57.6	54.7	56.1	69.3	42.3	52.5
	−PN	1	No	55.7	52.6	54.1	71.4	41.5	52.5
	−MWE	1	No	53.3	49.6	51.4	64.3	40.6	49.8
	−PN −MWE	1	No	51.8	47.9	49.8	65.6	38.3	48.4
		1+2	No	50.2	45.8	47.9	63.2	37.4	47.0
		1	Yes	52.1	48.3	50.1	65.7	38.2	48.3
		1+2	Yes	50.3	45.9	48.0	63.2	37.4	47.0

Table 1: Evaluation (in %) of the monolithic classifier (Mono) and suite of binary classifiers (Suite) in terms of precision (P), recall (R), and F-score (F). The majority class baseline is also presented. (The Token set is one of wordform feature set (W), lemma feature set (L), and the combined set (W+L); the lexical entries (LEs) under consideration were the entire set (All), or all except proper nouns (−PN), or all except MWEs (−MWE), or all except proper nouns or MWEs (−PN −MWE); the n-grams are either unigrams (1) or mixed unigrams and bigrams (1+2); and a given model optionally makes use of number folding (Numb)).

three feature sets: wordform features only (W), lemma features only (L), and both wordform and lemma features (W+L). The results are summarised under the Token heading in Table 1.

The F-score of the monolithic classifier and suite of binary classifiers increases from wordform features, to lemma features, to both. One reason for this is that there are simply more lemma features than wordform features (140 to 30 windows on average, for a given instance). This is particularly evident when we consider that the lemma features have higher recall but lower precision than the wordform features over the suite. We find that the wordform features are generally more accurate, allowing the classifier to tend to classify fewer instances with greater confidence, while noise gets introduced to the lemma feature classifier caused by folding derivational morphology in the lemmatised corpus. For example, features for *kemudi* "to drive" and *mengemudi* "driving" will be erroneously generated when considering *pengemudi* "driver" because they share a lemma, and verbal context usually does not reliably indicate the count classifier of the noun.

Despite the overgeneration caused by the lemma features, the combination of the wordform and lemma features leads to the classifier with the best F-score. The same holds true for our later experiments, so we only report results on the combination of feature sets below.

In addition to providing direct insight into the optimal feature representation for count classifier learning, these results represent the first extrinsic evaluation of the Baldwin and Awab Malay lemmatiser, and suggest that lemmatiser is working effectively.

4.2 Comparison of Data Sets

Next, we contrasted excluding the proper noun and MWE lexical entries from KAMI. We generated four data sets: with both proper noun and MWE entries (All), with proper nouns but without MWE (−MWE), without proper nouns but with MWE entries (−PN), and without proper noun or MWE entries (−PN −MWE). The precision, recall, and F-score for wordform and lemma features are shown in Table 1, under the LEs heading.

Removing either proper noun or MWE entries caused the F-score to drop. The effects are somewhat more subtle when considering the precision–recall trade-off for the suite of classifiers.

Excluding proper noun entries from the data set causes the precision of the suite of classifiers to rise, but the recall to fall. This indicates that the removed entries could be classified with high recall but low precision. Examining the proper noun entries from KAMI, we notice that most of them tend to be of the class *orang* (the person counter, as other semantic classes of proper noun do not tend to be counted: consider *?four New Zealands* in English) — this is the majority class for the entire data set. Removing majority class items tend to lower the baseline and make the task more difficult.

Excluding MWE entries from the data set causes both the precision and the recall of the suite to drop. This indicates that the removed entries could be classified with high precision and high recall, suggesting that they are generally compositional (e.g. *lap dog* would take the semantics of its head *dog* and is hence compositional, while *hot dog* is not). In KAMI, MWEs are often listed with the lemma of the semantic head; if the simplex head is an entry in the training data, then it becomes trivial to classify compositional MWEs without considering the noisy context features.

So, contrary to our intuition, proper nouns and MWEs tend to be easier to classify than the rest of the data set as a whole. This may not be true for other tasks, or even the general case of this task on unseen data, but the way we generate the data set from KAMI could tend to overestimate our results from these classes of lexeme. Consequently, we only describe results on the data set without proper nouns or MWEs below, to give a more conservative estimate of system performance.

4.3 Monolithic vs. Suite Classification

We consider the baseline for the task to be that of the majority class: *orang*. It corresponds to 1185 out of 2795 total unique headwords in the restricted data set from KAMI: baseline precision is 42.4%, recall is 39.3% when multiple headword–class assignments are considered. In Table 1 we present both the baseline performance over all lexical entries, and that over all lexical entries other than proper nouns and MWEs. Note that the baseline is independent of the classifier architecture (monolithic or binary suite), but duplicated across the two columns for ease of numerical comparison.

Both the monolithic classifier and suite significantly outperform the baseline in terms of F-score, the monolithic somewhat more so ($\chi^2 < 0.1, \mathrm{P} < 1$). Notably, *orang* entries made up about 40% of the data set, and the generic (i.e. the most semantically inspecific) count classifier *buah* made up about another 30% of the lexemes. It seemed that, since the 70% of the dataset covered by these two classes was greater than our performance, a classifier which could reliably distinguish between these two classes — while ignoring the other 49 classes — could have commensurately higher performance. (In effect, this would require distinguishing between persons for *orang* and things for *buah*, semantically speaking.) While the performance of that two-class system was greater for those classes under consideration, overall performance was slightly worse.

The suite of classifiers generally had higher precision and lower recall than the monolithic classifier. This indicates that the suite tended to make fewer positive assignments, but was more accurate when it did so. While the monolithic classifier was forced to make a classifier decision for every instance, the suite of classifiers tended to be conservative in classification, and consequently ended up with numerous headwords for which it did not predict a count classifier at all. This is particularly unsurprising when we consider that the prior probability for every category was less than 50%, so the default maximum likelihood is to define no headword as belonging to a given class.

4.4 Number features

As a further experiment, we examined the set of context windows where one of the tokens was numeric. It seems reasonable for a lexicographer to have defined the extent of numeric tokens ahead of time, given a moderately large corpus; numerals are of particular importance to the selection of a count classifier, in that they indicate the preference of how a given countable noun is counted.

We list the possible word-based numeral tokens below, which we combine with a numeric regular expression. Compound number formation is similar to English, and tokens beneath *sembilan* "nine" in the table are usually prefixed by *se-* as a replacement for a *satu* premodifier. (For example, *sebelas* instead of *satu belas* for "eleven".)

We attempted to classify after folding all tokens of this nature into a single "numeric" feature (Numb in Table 1), thereby allowing different numbers to be directly compared. The net effect was to raise

Token	Description
satu	"one"
dua	"two"
tiga	"three"
empat	"four"
lima	"five"
enam	"six"
tujuh	"seven"
lapan	"eight"
sembilan	"nine"
belas	"-teen"
puluh	"ten"
ratus	"hundred"
ribu	"thousand"
juta	"million"
bilion	"billion"
trilion	"trillion"

Table 2: Numbers in Malay, with English glosses.

the F-score of the monolithic classifier from 49.8% to 50.1%, and to lower the F-score of the suite of classifiers from 48.4% to 48.3%. Neither of these results are significantly different.

We also considered using a strategy more akin to what a human might go through in attempting to categorise nouns, in looking for all direct attestations of a given noun in the immediate context of a number expression, excluding all other lexical contexts (ideally being left with only contexts of the type "number CL noun", like *tiga biji pisang* above). However, the net effect was to remove most context windows, leaving numerous exemplars which could not be classified at all. Predictably, this led to an increase in precision but large drop in recall, the net effect of which was a drop in F-score. Folding numeric features here only improved the F-score by a few tenths of a percent.

Folding the numeric features turned out not to be valuable, as the fact that a given noun has numerals in its context is not a strong indicator of a certain type of count classifier, and all training examples were given an approximately equal benefit. Instead, we considered bi-word features, hoping to capture contexts of the type *tiga biji* "three CL", which seem to be a better indication of count classifier preference. A comparison of uni-word features and the combination of uni-word and bi-word features is shown in Table 1 under the n-gram heading.

The net effect of adding bi-word features is to reduce performance by one or two percent. One possible reason for this is the addition of many features: a context half-window of four tokens generates six more bi-word features.

Once more, excluding context windows without a numeral expression feature causes performance to drop drastically to 32.6% and 37.6% F-score for the monolithic classifier and suite, respectively. Folding numeric expression features again only changes figures by about one-tenth of one percent.

5 Discussion

We achieved about 50% F-score when attempting to predict the count classifiers of simplex common nouns in Malay. Although getting every second prediction incorrect seems untenable for automatic creation of a reliable lexical resource, we can see a system such as this being useful as a part of a semi-automatic system, for example, by providing high-confidence predictions of missing lexical entries to be judged by a human lexicographer. The precision of the suite of classifiers seems promising for this, having an error rate of about one in three predictions.

To our knowledge, there are no studies of Malay count classifiers to which we can directly compare these results. One possibility is to extend the work to Indonesian count classifiers — the mutual intelligibility and similar syntax seem amenable to this. However, it seems that there is a systematic divergence between count classifiers in these two languages: they are optional in many cases in Indonesian where they are not in Malay, and some counters do not map across well (e.g. *bilah*, the counter for knives and other long, thin objects, is not a count classifier in Indonesian). On the other hand, our approach does not rely on any features of Malay, and we expect that it would pick up surface cues equally well in Indonesian to achieve similar performance. The use of parsers like that of Gusmita and Manurung (2008) could provide further benefits.

When examining the feature sets we used, we discovered that features generated on the wordform level tended to be higher precision than features generated from the lemmatised corpus. One reason for this is that regular inflectional or derivational morphology is stripped to give numerous misleading contexts in the corpus. Consequently, we expect that some entries in the data set are added erroneously: where the wordform is not attested in the raw corpus and all of the instances of the lemma in the lemmatised corpus correspond to verbal or other derived forms. Given that almost a third of the data, or 945 lexemes, had no wordform instances, we might be underestimating our recall to some extent. With the aid of a POS tagger, we could condition the lemma windows in the feature set on a nominal POS for the candidate, and we would expect precision for these features to increase.

We observed that proper nouns and MWEs were less difficult to classify than the data set as a whole. Countable proper nouns tend to be people (e.g. *Australians* in English), and take *orang* as a counter, so for a given proper noun, we expect that the countable–uncountable cline to be the only hurdle to choosing the appropriate count classifier. MWEs tend to be compositional and hence share their class with their head, and an independent strategy for determining the head of the expression could transform such an entry into an easier simplex one.

The monolithic classifier, to choose between the entire set of 35 or so classes (after removing singleton classes), had a better F-score than the suite of binary classifiers, which predicted the suitability of a given count classifier one class at a time. However, the suite tended to be more conservative in its classifications, and had greater precision than the monolithic classifier. While high recall is useful in many circumstances, we feel that precision is more important as part of a pipeline with a lexicographer making gold-standard judgements. This trade-off makes combination of the two systems seem promising. One way could be to classify a given headword using both systems, and vote for the preferred class. Confidence of the classification, as given by the posterior probability of the maximum entropy model, could be taken into account. A two-tiered system could also be used, where the suite could be used to predict a high-precision class, and then the monolithic classifier could force a prediction for the entries where no classes at all were assigned by the suite. That case, where no classes are assigned, is particularly problematic with generation, for example, and other circumstances when a decision is required.[7]

Numeric features, we discovered, were only theoretically useful for bi-word contexts. Even then, we discerned little benefit when tailoring the feature vector to them. One possibility was that there was too much noise amongst the bigram features for the classifier to reliably use them as context. We feel, however, that restricting oneself to the syntax of numeral count expressions oversimplifies the problem somewhat. Since the preferred count classifier for a given noun tends to describe some underlying semantic property, the distributional hypothesis contends that surface cues from all contexts provide evidence. Our data agrees with this to some extent.

6 Conclusion

We developed a data set of labelled lexemes in Malay corresponding to headwords and count classifiers which were attested in a raw or lemmatised corpus. A maximum entropy classifier based on features generated from context tokens achieved about 50% F-score in prediction, and about 65% precision when a suite of binary classifiers was used. We envisage such a system to be a useful part of semi-automatic lexicography over a range of languages.

Acknowledgments

We would like to thank Francis Bond for his wisdom and advice throughout this research. NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme.

References

Mirna Adriani, Jelita Asian, Bobby Nazief, Seyed M. M. Tahaghoghi, and Hugh E. Williams. 2007. Stemming Indonesian: A confix-stripping approach. *ACM Transactions on Asian Language Information Processing*, 6:1–33.

Jelita Asian, Hugh E. Williams, and Seyed M. M. Tahaghoghi. 2004. A testbed for Indonesian text

[7]The issue of no classes being assigned is interesting when the countability–uncountability cline is considered. Potentially this could be an indication that the noun is, in fact, weakly countable. See, for example, Bond et al. (1994).

retrieval. In *Proceedings of the 9th Australasian Document Computing Symposium*, pages 55–58, Melbourne, Australia.

Timothy Baldwin and Su'ad Awab. 2006. Open source corpus analysis tools for Malay. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 2212–5, Genoa, Italy.

Timothy Baldwin and Francis Bond. 2003. Learning the countability of English nouns from corpus data. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 463–470, Sapporo, Japan.

Timothy Baldwin. 2007. Scalable deep linguistic processing: Mind the lexical gap. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation*, pages 3–12, Seoul, Korea.

Francis Bond and Kyonghee Paik. 1997. Classifying correspondence in Japanese and Korean. In *Proceedings of the 3rd Conference of the Pacific Association for Computational Linguistics*, pages 58–67, Tokyo, Japan.

Francis Bond and Kyonghee Paik. 2000. Reusing an ontology to generate numeral classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 90–96, Saarbrücken, Germany.

Francis Bond, Kentaro Ogura, and Satoru Ikehara. 1994. Countability and number in Japanese-to-English machine translation. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 32–38, Kyoto, Japan.

Dan Flickinger and Francis Bond. 2003. A two–rule analysis of measure noun phrases. In *Proceedings of the 10th International Conference on Head–Driven Phrase Structure Grammar*, pages 111–121, East Lansing, USA.

Raymund G. Gordon, Jr, editor. 2005. *Ethnologue: Languages of the World, Fifteenth Edition*. SIL International.

Ria Hari Gusmita and Ruli Manurung. 2008. Some initial experiments with Indonesian probabilistic parsing. In *Proceedings of the 2nd International MALINDO Workshop*, Cyberjaya, Malaysia.

Septina Dian Larasati and Ruli Manurung. 2007. Towards a semantic analysis of bahasa Indonesia for question answering. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 273–280, Melbourne, Australia.

Kyonghee Paik and Francis Bond. 2001. Multilingual generation of numeral classifiers using a common ontology. In *Proceedings of the 19th International Conference on the Computer Processing of Oriental Languages*, pages 141–147, Seoul, Korea.

Chiew Kin Quah, Francis Bond, and Takefumi Yamazaki. 2001. Design and construction of a machine-tractable Malay-English lexicon. In *Proceedings of the 2nd Biennial Conference of ASIALEX*, pages 200–205, Seoul, Korea.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 1–15, Mexico City, Mexico.

Kiyoaki Shirai, Takenobu Tokunaga, Chu-Ren Huang, Shu-Kai Hsieh, Tzu-Yi Kuo, Virach Sornlertlamvanich, and Thatsanee Charoenporn. 2008. Constructing taxonomy of numerative classifiers for Asian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 397–402, Hyderabad, India.

Virach Sornlertlamvanich, Wantanee Pantachat, and Surapant Meknavin. 1994. Classifier assignment by corpus-based approach. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 556–561, Kyoto, Japan.

Mashitah Taharin, editor. 1996. *Kamus Dewan Edisi Ketiga*. Dewan Bahasa Dan Pustaka, Kuala Lumpur, Malaysia.

Leonoor van der Beek and Timothy Baldwin. 2004. Crosslingual countability classification with EuroWordNet. In *Papers from the 14th Meeting of Computational Linguistics in the Netherlands*, pages 141–155, Antwerp, Belgium.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 124-132

Transforming Wikipedia into Named Entity Training Data

Joel Nothman and **James R. Curran** and **Tara Murphy**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{jnot4610,james,tm}@it.usyd.edu.au

Abstract

Statistical named entity recognisers require costly hand-labelled training data and, as a result, most existing corpora are small. We exploit Wikipedia to create a massive corpus of named entity annotated text. We transform Wikipedia's links into named entity annotations by classifying the target articles into common entity types (e.g. person, organisation and location). Comparing to MUC, CONLL and BBN corpora, Wikipedia generally performs better than other cross-corpus train/test pairs.

1 Introduction

Named Entity Recognition (NER), the task of identifying and classifying the names of people, organisations, locations and other entities within text, is central to many NLP tasks. The task developed from information extraction in the Message Understanding Conferences (MUC) of the 1990s. By the final two MUC evaluations, NER had become a distinct task: tagging the aforementioned proper names and some temporal and numerical expressions (Chinchor, 1998).

The CONLL NER evaluations of 2002 and 2003 (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) focused on determining superior machine learning algorithms and feature models for multilingual NER, marking tags for person (PER), organisation (ORG), location (LOC) and miscellaneous (MISC; broadly including e.g. events, artworks and nationalities). Brunstein (2002) and Sekine et al. (2002) expanded this into fine-grained categorical hierarchies; others have utilised the WordNet noun hierarchy (Miller, 1998) in a similar manner (e.g. Toral et al. (2008)). For some applications, such as biotextmining (Kim et al., 2003) or astroinformatics (Murphy et al., 2006), domain-specific entity classification schemes are more appropriate.

Statistical machine learning systems have proved successful for NER. These learn terms and patterns commonly associated with particular entity classes, making use of many contextual, orthographic, linguistic and external knowledge features. They rely on annotated training corpora of newswire text, each typically smaller than a million words. The need for costly, low-yield, expert annotation therefore hinders the creation of more task-adaptable, high-performance named entity (NE) taggers.

This paper presents the use of Wikipedia[1]—an enormous and growing, multilingual, free resource—to create NE-annotated corpora. We transform links between encyclopaedia articles into named entity annotations (see Figure 1). Each new term or name mentioned in a Wikipedia article is often linked to an appropriate article. A sentence introducing Ian Fleming's novel Thunderball about the character James Bond may thus have links to separate articles about each entity. Cues in the linked article about Ian Fleming indicate that it is about a person, and the article on Thunderball states that it is a novel. The original sentence can then be automatically annotated with these facts. Millions of sentences may similarly be extracted from Wikipedia to form an enormous corpus for NER training.

[1]http://www.wikipedia.org

Having produced annotated text in this manner, it can be used to train an existing NER system. By training the C&C tagger (Curran and Clark, 2003) on standard annotated corpora and Wikipedia-derived training data, we have evaluated the usefulness of the latter. We have used three gold-standard data sets, and have found that tagging models built on each perform relatively poorly on the others. Our Wikipedia-derived corpora are usually able to exceed the performance of non-corresponding training and test sets, by up to 8.7% F-score. Wikipedia-derived training data may also be more appropriate than newswire for many purposes.

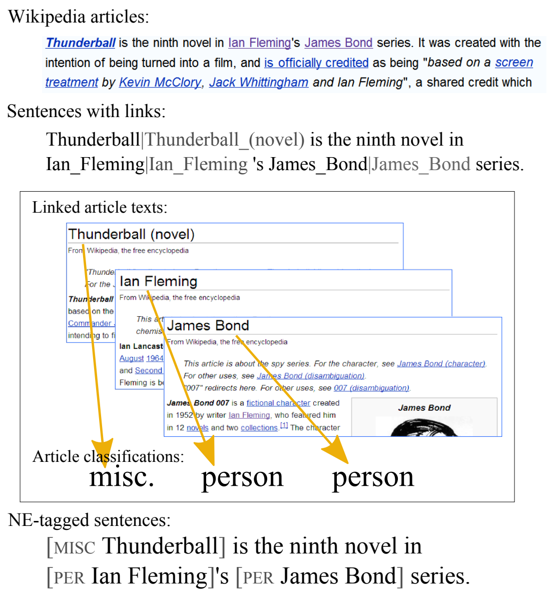
In a similar manner, free, large named entity-annotated corpora can be flexibly engineered for general or domain-specific tasks, allowing for NER without any manual annotation of text.

2 NER and Wikipedia

Following the CoNLL evaluations which focused on machine learning methods (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), work to improve NER performance has often involved the use of external knowledge. Since many tagging systems utilise categorised lists of known entities, some research has focused on their automatic extraction from the web (Etzioni et al., 2005) or Wikipedia (Toral et al., 2008), although Mikheev et al. (1999) and others have shown that larger NE lists do not necessarily correspond to increased NER performance. Nadeau et al. (2006) use such lists in an unsupervised NE recogniser, outperforming some entrants of the MUC Named Entity Task. Unlike statistical approaches which learn patterns associated with a particular type of entity, these unsupervised approaches are limited to identifying only common entities present in lists or those identifiable by hand-built rules.

External knowledge has also been used to augment supervised NER approaches. Kazama and Torisawa (2007) produced an F-score increase of 3% by including a Wikipedia-based feature in their NER system. Such approaches are nonetheless limited by the gold-standard data already available.

A less-common approach is the automatic creation of training data. An et al. (2003) extracted sentences containing listed entities from the web, and produced a 1.8 million word Korean corpus that gave similar results to manually-annotated training data. Richman and Schone (2008) used a method similar to that presented here in order to derive NE-annotated corpora in languages other than English. Their approach involves classifying English Wikipedia articles and using Wikipedia's inter-language links to infer classifications in other languages' articles. With these classifications they automatically annotate entire articles for NER training, and suggest that their results with a 340k-word Spanish corpus are comparable to 20k-40k words of gold-standard training data.



Thunderball is the ninth novel in Ian Fleming's James Bond series. It was created with the intention of being turned into a film, and is officially credited as being "*based on a screen treatment by Kevin McClory, Jack Whittingham and Ian Fleming*", a shared credit which

Thunderball|Thunderball_(novel) is the ninth novel in Ian_Fleming|Ian_Fleming 's James_Bond|James_Bond series.

[MISC Thunderball] is the ninth novel in [PER Ian Fleming]'s [PER James Bond] series.

Figure 1: Deriving training sentences from Wikipedia text: sentences are extracted from articles; links to other articles are then translated to NE categories.

3 From Wikipedia to NE corpora

Wikipedia is a multilingual online encyclopedia written by many thousands of its users, and includes over 2.3 million articles in English alone. We take advantage of Wikipedia's links between articles to derive a NE-annotated corpus. Since around 74% of Wikipedia articles (see Table 2) describe topics falling under traditional entity classes, many of Wikipedia's links correspond to entity annotations in gold-standard NER training corpora. These links also disambiguate their referent, distinguishing David Jones, a department store, from David Jones, a British poet. In summary, an entity-tagged corpus may be derived by the following steps (see Figure 1):

1. Classify all articles into entity classes
2. Split Wikipedia articles into sentences
3. Label NEs according to link targets
4. Select sentences for inclusion in a corpus

This same approach could be applied to multiple languages, or different granularities or domains of NE categories. We use the standard CoNLL categories (LOC, ORG, PER, MISC) in order to facilitate evaluation.

4 Classifying Wikipedia articles

In order to label links according to their targets, we first must classify Wikipedia's articles into a fixed set of entity categories.

Many researchers have already tackled the task of classifying Wikipedia articles, often into named entity categories. The current state-of-the-art results (90% F-score) were achieved using bag-of-words with an SVM learner (Dakka and Cucerzan, 2008). They also make use of entities co-occurring in lists as a classification heuristic. While Wikipedia provides its own categorisation hierarchy, it has been described as a *folksonomy*, comparable to other collaborative online tagging. Suchanek et al. (2007) divide Wikipedia categories into conceptual (Sydney is a coastal city in Australia), relational (Ian Fleming had a 1908 birth), thematic (James Bond has theme James Bond) and administrative (the Sydney article is available as a spoken article). Conceptual categories, which are most useful for categorisation, often have plural head nouns (e.g. cities of Coastal cities in Australia) which describe the nature of member articles.

We use a bootstrapping approach to classification (see Figure 2), with heuristics based primarily on category head nouns and definitional opening sentences of articles. This approach reflects our intuitions that: (a) it is difficult to manually design rules with high coverage; (b) bag-of-words methods lose structural and linguistic information; (c) a semi-supervised approach is able to learn heuristics from unlabelled data; (d) we may easily leave an article's class undecided and ignore it in future processing.

Each article is classified as one of: unknown (UNK; not a target category for evaluation, like O in IOB tagging); a member of a NE category; a disambiguation page (DAB; these list possible referent articles for a given title); or a non-entity (NON). We identify these final two categories largely on the basis of specialised static heuristics, while entity classes are assigned through mappings learnt in the bootstrapping process.

4.1 Non-entity heuristics

Because of the diversity of non-entity articles, and the ease of identifying large portions of them, we first attempt to classify each article as a non-entity. We generally assume that articles whose incoming links are largely lowercase are non-entities, and also classify as NON all articles with a title beginning List of , together finding 32% of NON articles in our hand-labelled data. We also separately identify DAB articles on the basis of their title and categories.

4.2 Bootstrapped heuristics

For general classification, we extract features from articles, which may each be mapped to an entity class. These mappings are produced by the bootstrapping process.

Category nouns Using the C&C tools, we POS tagged and chunked (shallow phrasal parsing) all category titles in Wikipedia in order to determine their head nouns, the last word of the first noun phrase chunk. If the POS tagger identified this head as plural, we assume that the category is *conceptual* by Suchanek et al.'s (2007) designation. Thus institutions would be extracted as the head of category Educational institutions established in 1850 and might identify the category constituents as belonging to ORG. Each such phrasal head, or bigram collocation (differentiating radio stations from railway stations), is considered a feature which may be mapped to an entity class. The most frequent class present among an article's categories is then assigned to the article. If no conceptual categories with mappings are identified, or multiple classes tie maximum, UNK is tentatively assigned.

Definition nouns Where category nouns are inconclusive, or the maximum category class only

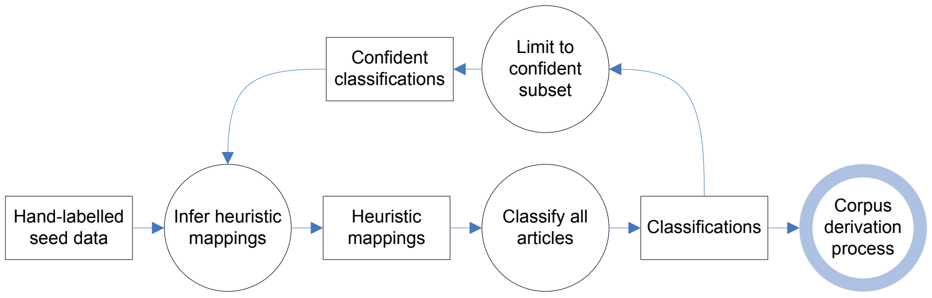


Figure 2: A bootstrapping approach to article classification

leads by 1, we resort to a secondary heuristic feature. Kazama and Torisawa (2007) make the assumption that many articles' opening sentences are in the form of definitions, and hence the noun phrase following a copula (is, are, was, were) is indicative of the article's category. Thus the article on Sydney opens Sydney is the most populous city in Australia..., wherein the word city best identifies the class of the article as LOC. We extract only one such definition noun (again, a unigram or bigram collocation) by POS tagging and chunking the first sentence of each article. Like with category nouns, this may be mapped to an entity class which relabels articles previously marked UNK, or in case of contradicting heuristics may revert a classification to UNK.

4.3 Bootstrapping

As shown in Figure 2, bootstrapping in our classification process involves using hand-labelled data to initialise mappings from category and definition nouns to entity classes, and having a feedback loop in which we use the confident results of one classification to produce heuristic mappings for the next. Since most articles fall within multiple categories, each subsequent bootstrap produces new mappings (until convergence), and therefore allows the confident classification of a larger portion of Wikipedia.

We infer mappings as follows: Given a set of articles and their classes, we can count the number of times each feature occurs with the class. For each candidate noun N (unigram or bigram), the class k with which it is most often associated is determined. If n classified articles support the mapping $N \rightarrow k$ and m articles contradict it, then we accept the mapping if $n \geq t$ and $\frac{m}{n+m} < p$, for some constant thresholds t and p. We have used $p = .25$, with values for t given in Table 1.

t	Category NNs	Definition NNs
Seed inference	1	2
Feedback	2	4

Table 1: Varying threshold values for inference.

An article classification is considered confident for use in bootstrapping if it is not labelled UNK, and if none of the heuristic features disagree (i.e. all category and definition features available map to the same class).

A single annotator manually labelled 1300 Wikipedia articles with over 60 fine-grained category labels based on Brunstein (2002), which were reduced for evaluation to {LOC,PER,ORG,MISC,NON,DAB}, apart from 7 ambiguous entities which were left unlabelled. Initially 1100 random articles were labelled, but we found that the data poorly represented popular entity types with few instances, such as countries. Hence an additional 200 articles were randomly selected from among the set of articles with over 700 incoming links. The distribution of classes in the data is shown in Table 2. Of this set, we used 15% in a held-out test set for development, and performed final evaluation with ten-fold cross-validation.

4.4 Classification results

We found that evaluation statistics stabilised for our held-out test set after three bootstrap loops. Although even initial mappings (1050 category nouns; 132 definition nouns) produced a micro-averaged F-score of 84%, this reflects the high proportion of easily-identifiable PER articles, as the macro-average was much lower (63%). After the third bootstrap, with 8890 category nouns and 26976 definition nouns, this had increased to 91% micro- and 90% macro-average F-score. 12% of

Class	%	*P*	*R*	*F*
LOC	19	95	94	95
PER	24	96	98	97
ORG	14	92	80	85
MISC	18	93	72	80
DAB	5	100	93	96
NON	20	89	69	78
All		94	84	89
Entities only		96	88	92

Table 2: The class distribution within our manual article labels and average results of a ten-fold cross-validation. Overall results are micro-averaged.

the test data was labelled UNK.

Per-class and overall results of cross-validation are shown in Table 2. Our largest failures are in recall for MISC and NON, by far the broadest classes and hence difficult to capture completely.

5 Extracting and selecting sentences

Wikipedia's articles are composed using a structural markup language specific to its software. While marked-up data is available, it requires cleaning, separation into sentences and tokenisation in order to be transformed into a NER training corpus. We produce a parse tree of the markup using `mwlib`[2], remove most non-sentential data and all markup other than inter-article links, and split article texts into sentences using Punkt (Kiss and Strunk, 2006)—an unsupervised algorithm for sentence boundary detection, trained here on Wikipedia data—before tokenising.

We need to select sentences for inclusion in our training corpus for which we are confident of having correctly labelled all named entities. For the generic NER task in English, this depends highly on capitalisation information. For instance, we simply might accept only sentences where all capitalised words have links to articles of known classification (not UNK or DAB). This criterion is overly restrictive: (a) it provides a low recall of sentences per article; (b) it is biased towards short sentences; and (c) since each entity name is often linked only on its first appearance in an article, it is more likely to include fully-qualified names than shorter referential forms (surnames, acronyms, etc.) found later in the article. We are also challenged by many words that are capitalised by English convention but do not correspond to entities. These and related problems are tackled in the following sub-sections.

5.1 Inferring additional links

In order to increase our coverage of Wikipedia sentences, we attempt to infer additional links. In particular, since Wikipedia style dictates that only the first mention of an entity should be linked in each article, we try to identify other mentions of that entity in the same article. We begin by compiling a list of alternative titles for each article. Then for any article in which we are attempting to infer links we produce a trie containing the alternative titles of all outgoing links. When a word with an uppercase letter is found, we find the longest matching string within the trie and assign its class to the matching text.

Alternative titles for an article A include:

Type 1 The title of A and those of redirects[3] to A (with expressions following a comma or within parentheses removed);

Type 2 The first or last word of A's title if A is of class PER;

Type 3 The text of all links whose target is A.

We have switched use of each type of inferred titles on and off in our experiments below.

5.2 Conventional capitalisation

As well as proper names, first words of sentences, pronouns (I in English), dates, adjectival forms of names (e.g. nationalities), personal titles and acronyms are capitalised in English. As an exception to our general sentence selection criterion, we include such capitalised words in our corpus.

First words If a word beginning a sentence or following some punctuation (semicolon, left-quote, etc.) is capitalised and unlinked, it may be difficult to determine whether it should be labelled as belonging to an entity. Unless an entity link can be inferred, a first word is ignored if it is found on a list of 1520 words (compiled from Wikipedia data) including collocational frequent sentence starters (Kiss and Strunk, 2006),

[2]A Python-based parser for MediaWiki markup. `http://code.pediapress.com`

[3]Redirect pages make articles accessible through alternative titles.

and words which are commonly both sentence-initial and lowercase when sentence-internal.

Dates Names of months and days of the week are identified by regular expressions.

Personal titles Personal titles (e.g. Brig. Gen., Prime Minister-elect) are conventionally capitalised in English. In some cases, such titles are linked to relevant articles, but e.g. U.S. President is in categories like Presidents of the United States, causing its incorrect classification as PER. We have implemented a trivial solution to catch some titles: if a link appears immediately before a link to a PER target, we assume that it is a title and may be included in the corpus without a NE tag. Note that this fails to handle the same titles when linked freely in text. We use the BBN corpus (Weischedel and Brunstein, 2005) to compile a list of titles that are rarely linked (e.g. Mr.).

Adjectival forms Adjectival forms of entity names, such as American or Islamic, are capitalised in English. While these are not technically entities, both the CoNLL and BBN gold-standard corpora (see section 6.1) tag them. Our rudimentary solution for this involves POS tagging the potential corpus text and relabelling entities as MISC if their final word is tagged as an adjective. This does not cover nationalities used as nouns, for which we currently retain the incorrect label.

5.3 Anomalous capitalisation

Capitalised non-entity links and all-lowercase entity links may be problematic. The former often results from mis-classification, or includes an NE in its title, e.g. Greek alphabet or Jim Crow laws, in which case it would be incorrect to leave the reference untagged. Lowercase entity links result from non-proper noun references to entities, e.g. in In the Ukraine, anarchists fought in the civil war ..., the text civil war links to Russian Civil War. Sentences with capitalised NON links or lowercase entity links are therefore discarded, excepting entities like gzip where Wikipedia marks the article as having a lowercase title.

5.4 Adjusting link boundaries

Link text sometimes incorporates more than just the entity name, such the possessive 's at the end of a name, or the linking of Sydney, Australia which should be treated as two separate entities. Hence we unlink the following strings when found at the end of link text: parenthesised expressions; text following a comma for LOC, ORG and PER; possessive 's; or other punctuation.

6 Evaluation

We evaluate our corpora by training the C&C tagger[4] to build separate models (a) when trained with Wikipedia data; (b) when trained with hand-annotated training data; (c) when trained with both combined, and comparing the tagging results on gold-standard test data. We use the C&C Maximum Entropy NER tagger with default orthographic, contextual, in-document and first name gazetteer features (Curran and Clark, 2003). Our results are given as per-category and micro-averaged phrasal precision, recall and F_1-score.

6.1 Gold-standard corpora

We evaluate our generated corpora against three sets of manually-annotated data from (a) the MUC-7 Named Entity Task (MUC, 2001); (b) the English CoNLL-03 Shared Task (Tjong Kim Sang and De Meulder, 2003); (c) the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005). Stylistic and genre differences between the source texts affect compatibility for NER, e.g. the CoNLL corpus formats headlines in all-caps, and includes much non-sentential data, such as tables of sports scores.

Each corpus uses a different set of entity labels, with MUC marking locations, organisations and personal names in addition to numerical and time information. CoNLL labels only proper names but adds a MISC category for all entities not otherwise tagged. BBN ambitiously annotate the entire Penn Treebank corpus with 105 fine-grained tags: 54 corresponding to CoNLL entities; 21 for numerical and time data; and 30 for other classes of terms. For the present evaluation, BBN's tags were reduced to the equivalent CoNLL tags, with non-CoNLL tags in the BBN and MUC data removed. Since no MISC entities are marked in MUC, such labels need to be removed from CoNLL, BBN and Wikipedia data for comparison.

[4] http://svn.ask.it.usyd.edu.au/trac/candc

Corpus	# tags	Number of tokens		
		TRAIN	DEV	TEST
MUC-7	3	84051	18764	60872
CONLL-03	4	203621	51362	46435
BBN	54	901894	142218	129654

Table 3: Corpora used for evaluation

All corpora were transformed into a common format and tagged with parts of speech using the Penn Treebank-trained (sections 2-21) C&C POS tagger. While standard training (TRAIN), development (DEV) and final test (TEST) set divisions were available for the CONLL and MUC data, the BBN corpus was split at our discretion: sections 03–21 for TRAIN, 00–02 for DEV and 22-24 for TEST. The corpus sizes are compared in Table 3.

6.2 Wikipedia data and experiments

Wikipedia's article text is made freely available for download.[5] We have used data from the 22 May 2008 dump of English Wikipedia which includes 2.3 million articles. Splitting this into sentences and tokenising produced 32 million sentences each containing an average of 24 tokens.

Our experiments were mostly performed with Wikipedia-derived corpora of 150,000 sentences. Despite having 14 million sentences available, we were limited by time and memory for training.

We report results from four groups of experiments: (a) how does a Wikipedia-trained tagger compare to gold-standard data? (b) what is the effect of training a tagger with both gold-standard and Wikipedia-derived data? (c) how does the number of sentences in the Wikipedia corpus affect performance? (d) to what extent does the inference of additional links (see section 5.1) affect results? Levels of link inference are differentiated between corpora WP0 (no inference), WP1 (type 1 alternative titles), WP2 (types 1–2) and WP3 (types 1–3). The 150k-sentence corpora contain 3.5 million tokens on average.

7 Results and discussion

As shown in Tables 4 and 5, each set of gold-standard training data performs much better on corresponding evaluation sets (italicised) than on test sets from other sources. The exception is for BBN on MUC TEST, due to differing TEST and

[5]`http://download.wikimedia.org/`

Training corpus	DEV overall F-score		
	MUC	CONLL	BBN
MUC	*83.4*	54.8	59.7
CONLL	64.5	*86.9*	60.2
BBN	75.0	58.0	*88.0*
WP0 – no inference	63.4	63.6	56.6
WP1	65.3	65.4	58.6
WP2	**68.1**	67.0	**60.6**
WP3 – all inference	64.4	**68.5**	58.0

Table 4: DEV results without MISC.

TRAIN	With MISC		No MISC		
	CONLL	BBN	MUC	CONLL	BBN
MUC	—	—	*74.4*	51.7	54.8
CONLL	*81.2*	62.3	58.8	*82.1*	62.4
BBN	54.7	*86.7*	75.7	53.9	*88.4*
WP2	58.9	62.3	67.5	60.4	58.8

Table 5: TEST results for WP2.

DEV subject matter. The 12-33% mismatch between training and evaluation data suggests that the training corpus is an important performance factor (see also Ciaramita and Altun (2005)).

A key result of our work is that the performance of non-corresponding hand-annotated corpora is often exceeded by Wikipedia-trained models.

We also assess using our Wikipedia corpora together with gold-standard data on traditional train-test pairs. Table 6 shows that this approach leads to only marginal variations in performance.

Table 4 also illustrates the effectiveness of link inference, which is able to increase F-score by 5%. Performance increases as inference types 1 (identifying article titles) and 2 (single words of PER titles) are added, but 3 (all incoming link labels) degrades performance on the MUC and BBN corpora, since it likely over-generates alternative titles. Matching the longest string may also falsely include additional words, e.g. tagging Australian citizen rather than Australian, to which inference level 3 is most susceptible.

In Figure 3, we illustrate the effect of varying the size of the Wikipedia-derived training data. Increased training data tends to improve performance to a point (around 25k sentences for MUC and 125k for BBN) after which improvements are marginal and results may degrade. The late stabilising of BBN performance is possibly caused by the size and breadth of its evaluation data sets.

To analyse overall error, our per-class results

Training corpus	TEST overall F-score		
	MUC	CONLL	BBN
Corresponding TRAIN	74.4	82.1	88.4
TRAIN + WP2	76.8	81.8	87.7

Table 6: Wikipedia as additional training data

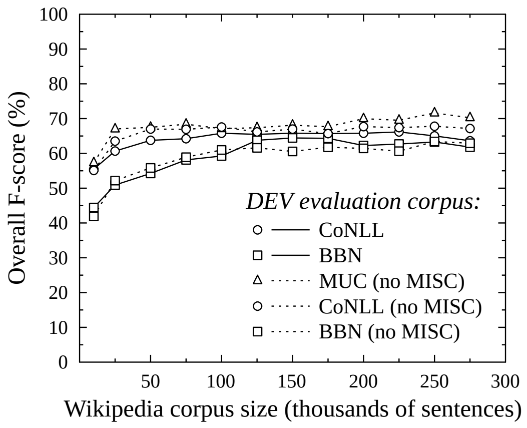


Figure 3: The effect of varying WP2 corpus size.

are shown in Table 7. LOC and PER entities are relatively easy to identify, although a low precision for PER suggests that many other entities have been marked erroneously as people, unlike the high precision and low recall of ORG. As an ill-defined category, with uncertain mapping between BBN and CONLL classes, MISC precision is unsurprisingly low. We also show results evaluating the correct labelling of each token, and the much higher results (13%) reflects a failure to correctly identify entity boundaries. This is common in NER, but a BBN-trained model only gives 5% difference between phrasal and token F-score. We believe this reflects Wikipedia links often including other tokens along with proper names.

Among common tagging errors we have identified, we find: tags continuing over additional words as in New York-based Loews Corp. all being marked as a single ORG; nationalities marked as LOC rather than MISC; White House a LOC rather than ORG, as with many sports teams; single-word ORG entities marked as PER; titles such as Dr. included in PER tags; untagged title-case terms and tagged lowercase terms in the gold-standard.

Our results suggest many avenues for improving corpus derivation, but highlight Wikipedia as a source of competitive training data.

Class	By phrase			By token		
	P	R	F	P	R	F
LOC	62.0	76.8	68.7	61.0	82.4	70.1
MISC	43.5	55.7	48.8	42.5	59.3	49.5
ORG	76.8	53.4	63.0	87.9	65.3	74.9
PER	48.0	81.0	60.3	56.3	95.5	70.8
All	60.9	63.8	62.3	76.7	72.5	74.6

Table 7: Results for each entity category when the WP2 model was evaluated on the BBN TEST set.

8 Conclusion and future work

There is much room for improving the results of our Wikipedia-based NE annotations. A more careful approach to link inference may reduce incorrect boundaries of tagged entities. Adding disambiguation pages as another source of alternative article titles will make the labelling of acronyms more common. Better labelling of personal titles and adjectival entity names may also provide great gain, as did our simple approaches.

Since the training corpus is not often a variable in NER research, we need to explore ways to fairly evaluate corpus-based experiments: the number of articles, sentences, tagged entities or tokens may all be chosen as variables or invariants in experiments; and differing genres or annotation schemes make results difficult to compare.

We have nonetheless shown that Wikipedia can be used a source of free annotated data for training NER systems. Although such corpora need to be engineered specifically to a desired application, Wikipedia's breadth may permit the production of large corpora even within specific domains. Focusing on this flexibility, we intend to experiment with finer-grained NE hierarchies, domain-specific annotation schema, and multilingual NER. Our results indicate that Wikipedia data can perform better (up to 8.7% for MUC on CONLL) than training data that is not matched to the evaluation, and hence is widely applicable. Transforming Wikipedia into training data thus provides a free and high-yield alternative to the laborious manual annotation required for NER.

Acknowledgments

We would like to thank members of the Language Technology Research Group and the anonymous reviewers for their helpful feedback. Curran and Murphy were funded under ARC Discovery grant

DP0665973 and Nothman was supported by a University of Sydney Honours Scholarship.

References

Joohui An, Seungwoo Lee, and Gary Geunbae Lee. 2003. Automatic acquisition of named entity tagged corpus from world wide web. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 165–168.

Ada Brunstein. 2002. Annotation guidelines for answer types. LDC2005T33.

Nancy Chinchor. 1998. Overview of MUC-7. In *Proceedings of the 7th Message Understanding Conference*.

Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*.

James R. Curran and Stephen Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 164–167.

Wisam Dakka and Silviu Cucerzan. 2008. Augmenting Wikipedia with named entity tags. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 545–552, Hyderabad, India.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32:485–525.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–8, Bergen, Norway.

George A. Miller. 1998. Nouns in WordNet. In *WordNet: An Electronic Lexical Database*, chapter 1. MIT Press.

2001. *Message Understanding Conference (MUC) 7*. Linguistic Data Consortium, Philadelphia.

Tara Murphy, Tara McIntosh, and James R. Curran. 2006. Named entity recognition for astronomy literature. In *Proceedings of Australian Language Technology Workshop*, pages 59–66, Sydney, Australia.

David Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, volume 4013 of *LNCS*, pages 266–277.

Alexander E. Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1–9, Columbus, Ohio.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended named entity hierarchy. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1818–1824.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: a core of semantic knowledge — unifying WordNet and Wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 142–147.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 1–4.

Antonio Toral, Rafael Muñoz, and Monica Monachini. 2008. Named entity WordNet. In *Proceedings of the 6th International Language Resources and Evaluation Conference*.

Ralph Weischedel and Ada Brunstein. 2005. *BBN Pronoun Coreference and Entity Type Corpus*. Linguistic Data Consortium, Philadelphia.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 133-141

Fit it in but say it well!

Cécile Paris, Nathalie Colineau, Andrew Lampert
CSIRO – ICT Centre
Locked Bag 17
North Ryde, NSW 1670, Australia
FirstName.LastName@csiro.au

Joan Giralt Duran
Barcelona School of Informatics
Technical University of Catalonia
Barcelona, Spain
joangi@gmail.com

Abstract

Reasoning about how much content to generate when space is limited presents an increasingly important challenge for generation systems, as the diversity of potential delivery channels continues to grow. The problem is multi-facetted: the generated text must fit into the allocated space, the space available must be well utilised, and the resulting text must convey its intended message, and be coherent, well structured and balanced. To address this problem, we use a discourse planning approach. Our system reasons about the discourse structure to decide how much content to realise. In this paper, we present two algorithms that perform this reasoning and analyse their effectiveness.

1 Introduction

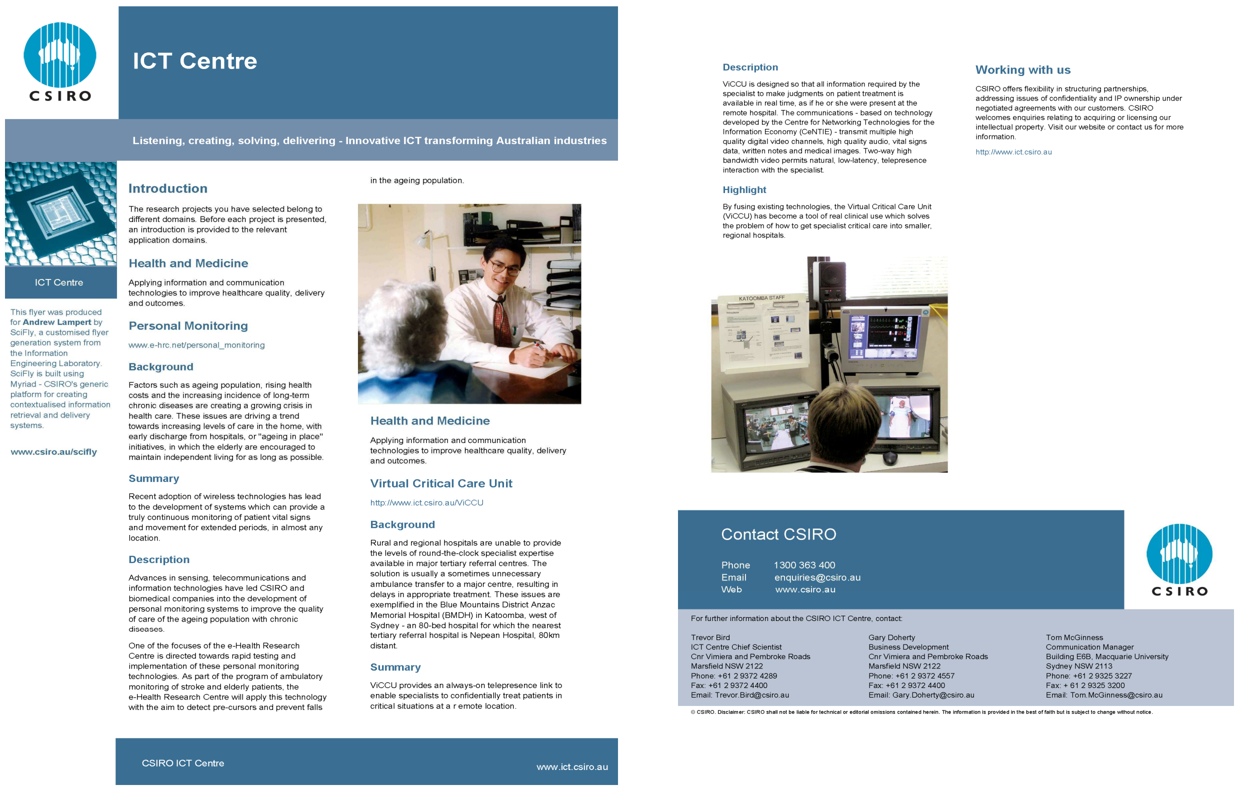
The ability to reason about how much content to realise in order to convey a message when the allocated space is fixed is an important consideration for Natural Language Generation (NLG) systems. It will become even more pressing as the amount of available information increases (e.g., via the web or content management systems) and the space constraints on the delivery media become more diverse (e.g., via web browsers, email, PDAs, cell phones).

We, as humans, address this problem by shortening our sentences or by restricting the content we include. We can achieve the former by manipulating vocabulary and syntax. This, however, is of limited value in reclaiming significant amounts of space and requires careful attention to sentence-level grammar and vocabulary choice. We can achieve the latter by dropping those pieces of content whose contribution to the communicative goal is most limited. For instance, we might question the need for a long elaboration or an example to illustrate our point. This approach can reclaim significant amounts of space but requires an understanding of the text's discourse structure.

Many organisations today store information in content management systems or other types of organisational databases. This information is typically stored at the level of paragraphs, along with images and database entries (e.g., directories of contact information). Such information systems provide an interesting opportunity for NLG systems, by reducing the cost of acquiring the underlying semantic knowledge base and removing the need to construct text from first principles. However, they also present challenges: how can a system still perform the type of reasoning typically expected of NLG systems when it has no control over the text at the sentence level? In particular, how can a system produce a coherent and well structured text that meets some specific space limitations? In these cases, a system only has one method available to ensure it produces an appropriate amount of text: it must reason about the text at the discourse level.

Many of our application domains are such that information is stored at the paragraph level in one or more data repositories. Our system answers people's information needs by retrieving appropriate pre-authored text fragments from such repositories and delivering that content via a variety of media, each with their own space requirements. The specific application we discuss in this paper is SciFly, a system that generates brochures about the work carried out in our organisation. The output is delivered as a two-page flyer, as shown in Figure 1[1], and can also be displayed as a web output and as a plain text summary in the body of an email. As our underlying text fragments are pre-authored, we cannot manipulate the realisation of their sentences. We thus concentrate on reasoning about the discourse structure to generate text fitting specific space constraints. Importantly, the brochures we

[1] Though the text is too small to read, the figure gives an idea of the document and its layout.



ICT Centre

Listening, creating, solving, delivering - Innovative ICT transforming Australian industries

ICT Centre

This flyer was produced for **Andrew Lampert** by SciFly, a customised flyer generation system from the Information Engineering Laboratory. SciFly is built using Myriad - CSIRO's generic platform for creating contextualised information retrieval and delivery systems.

www.csiro.au/scifly

Introduction

The research projects you have selected belong to different domains. Before each project is presented, an introduction is provided to the relevant application domains.

Health and Medicine

Applying information and communication technologies to improve healthcare quality, delivery and outcomes.

Personal Monitoring

www.e-hrc.net/personal_monitoring

Background

Factors such as ageing population, rising health costs and the increasing incidence of long-term chronic diseases are creating a growing crisis in health care. These issues are driving a trend towards increasing levels of care in the home, with early discharge from hospitals, or "ageing in place" initiatives, in which the elderly are encouraged to maintain independent living for as long as possible.

Summary

Recent adoption of wireless technologies has lead to the development of systems which can provide a truly continuous monitoring of patient vital signs and movement for extended periods, in almost any location.

Description

Advances in sensing, telecommunications and information technologies have led CSIRO and biomedical companies into the development of personal monitoring systems to improve the quality of care of the ageing population with chronic diseases.

One of the focuses of the e-Health Research Centre is directed towards rapid testing and implementation of these personal monitoring technologies. As part of the program of ambulatory monitoring of stroke and elderly patients, the e-Health Research Centre will apply this technology with the aim to detect pre-cursors and prevent falls in the ageing population.

Health and Medicine

Applying information and communication technologies to improve healthcare quality, delivery and outcomes.

Virtual Critical Care Unit

http://www.ict.csiro.au/ViCCU

Background

Rural and regional hospitals are unable to provide the levels of round-the-clock specialist expertise available in major tertiary referral centres. The solution is usually a sometimes unnecessary ambulance transfer to a major centre, resulting in delays in appropriate treatment. These issues are exemplified in the Blue Mountains District Anzac Memorial Hospital (BMDH) in Katoomba, west of Sydney - an 80-bed hospital for which the nearest tertiary referral hospital is Nepean Hospital, 80km distant.

Summary

ViCCU provides an always-on telepresence link to enable specialists to confidentially treat patients in critical situations at a r emote location.

CSIRO ICT Centre www.ict.csiro.au

Description

ViCCU is designed so that all information required by the specialist to make judgments on patient treatment is available in real time, as if he or she were present at the remote hospital. The communications - based on technology developed by the Centre for Networking Technologies for the Information Economy (CeNTIE) - transmit multiple high quality digital video channels, high quality audio, vital signs data, written notes and medical images. Two-way high bandwidth video permits natural, low-latency, telepresence interaction with the specialist.

Highlight

By fusing existing technologies, the Virtual Critical Care Unit (ViCCU) has become a tool of real clinical use which solves the problem of how to get specialist critical care into smaller, regional hospitals.

Working with us

CSIRO offers flexibility in structuring partnerships, addressing issues of confidentiality and IP ownership under negotiated agreements with our customers. CSIRO welcomes enquiries relating to acquiring or licensing our intellectual property. Visit our website or contact us for more information.

http://www.ict.csiro.au

Contact CSIRO

Phone 1300 363 400
Email enquiries@csiro.au
Web www.csiro.au

For further information about the CSIRO ICT Centre, contact:

Trevor Bird	Gary Doherty	Tom McGinness
ICT Centre Chief Scientist	Business Development	Communication Manager
Cnr Vimiera and Pembroke Roads	Cnr Vimiera and Pembroke Roads	Building E6B, Macquarie University
Marsfield NSW 2122	Marsfield NSW 2122	Sydney NSW 2113
Phone: +61 2 9372 4289	Phone: +61 2 9372 4557	Phone: +61 2 9325 3227
Fax: +61 2 9372 4400	Fax: +61 2 9372 4400	Fax: + 61 2 9325 3200
Email: Trevor.Bird@csiro.au	Email: Gary.Doherty@csiro.au	Email: Tom.McGinness@csiro.au

© CSIRO. Disclaimer: CSIRO shall not be liable for technical or editorial omissions contained herein. The information is provided in the best of faith but is subject to change without notice.

Figure 1: A brochure generated by our system

generate need to look professional, as if manually written. Coherence and good structure is paramount.

In the remainder of this paper, we discuss other work which has looked at controlling the amount of generated text. We then present our approach to generation when a text has to fit into some specific space and the system has no control over the text at the sentence level. We show how we can exploit the discourse structure to decide how much content to realise. We present two algorithms that perform this reasoning and analyse their comparative performance.

2 Related work

Generation systems have often exploited the discourse structure for a number of reasoning tasks. To achieve this, systems build a discourse tree during discourse planning. The tree includes the communicative goals that were achieved and the rhetorical relations holding between text spans, frequently represented using Rhetorical Structure Theory (RST) (Mann and Thompson, 1988). The systems then reason about this discourse tree to allow them to participate in a dialogue (e.g., Moore and Swartout, 1989), generate appropriate cue phrases to link two spans of text (e.g., Scott and de Souza, 1990) or reason about layout (e.g., Bateman et al., 2001). See Taboada and Mann (2006) for other applications.

Our system uses discourse trees to reason about how much content to express in order to fill some specific available space. Other systems have performed reasoning at the discourse structure level to control how much text is generated. Moore and Paris (1993), for example, allowed their discourse planner to be set in *terse* or *verbose* mode to produce short or long texts. Their approach thus constrained the length of the generated content at the onset of the process. However, this can be overly restrictive. In contexts such as ours, for example, in which similar content must be delivered via several media, it is desirable to produce one discourse tree that can then be delivered appropriately to the different delivery channels and conform to different space constraints.

During discourse planning, our system specifies the RST relations that hold between the retrieved pre-authored text fragments and exploits the RST principle of nuclearity to shorten a text. In RST, a relation holds between the *nucleus*, usually considered the main information to be conveyed, and a *satellite*, which provides supporting information.[2] The intuition is that nuclei are more important while satellites can be dropped. Systems that use the discourse structure to produce summaries are also based on this intuition (e.g., Spark-Jones, 1993; Ono et al., 1994; Rino and Scott, 1994; Marcu, 1998). However, while techniques used in

[2] There are a few exceptions for multinuclear relations.

both approaches are similar, the purpose and challenges of the two tasks are different: in a summary, one wishes to convey only the most essential part of a message, leaving out all the content that could be considered optional. In our task, we want to produce a text that fits the available space. In some cases, the original content is only, let's say, one paragraph too long to fit into the required space. The issue is to find the least essential paragraph to convey the message and still obtain a balanced text, as opposed to summarising the text per se. This is akin to the distinctions between shortening a paper by ½ a page to fit into the required 8-page limit and writing its abstract. In addition, shortening a text (rather than summarising it) raises a new challenge: that of ensuring that the final text is still balanced. Consider for example cases in which there are bulleted items. When shortening the text, one needs to ensure that all the items remain at about the same length, or the text will appear odd.

Our approach exploits the semantics of the rhetorical relations and the notion that some relations are more important than others. O'Donnell (1997) used the same principle to assign relevance scores to text nodes in the discourse structure to produce documents of variable length. While our approach is similar to O'Donnell's in that respect, his approach required individual sentences to be manually marked up with rhetorical relations. This allowed his system to manipulate the text at or below the sentence level, although repair had to occur after the process to ensure coherence and grammaticality. O'Donnell's approach was thus close to traditional NLG systems that build text from first principles and are able to vary the amount of text at the lexico-grammatical level (e.g., Reiter, 2000).

Like most NLG-based approaches to constraining the length of a text, we use greedy algorithms to cut content. Vander Linden (2008) reports on an alternate approach that used dynamic programming. His work has not yet been evaluated, so it is unclear how valuable it could be in our context.

3 System Architecture

To achieve our goal of delivering tailored and coherent information, we build upon both NLG and document synthesis technology to retrieve and repurpose information (Colineau *et al.*, 2004). The retrieval process is orchestrated by a discourse planner that, using discourse plans, builds a discourse tree specifying what to extract, for which purpose and how the various pieces of information relate to each other (Paris *et al.*, 2008).

We use SciFly to illustrate how our system reasons about space constraints. Given a query from a user (one or more topic(s) of interest), SciFly consults a repository of text fragments to assemble the relevant information. The fragments, written by the marketing team of our organisation, are self contained and comprised of typically one paragraph, two at most.[3] SciFly integrates all the relevant fragments into a coherent whole (see Figure 1) using meta-data describing each fragment. The meta-data chosen is that envisaged for a new content management system for our organisation's website. Note that this meta-data does not correspond to rhetorical relations. It is the discourse plans that later determine which relation holds between fragments given their role in a specific communicative goal. As a result, a text fragment can be used with different relations in different brochures.

SciFly also produces a web output, and a PDF version of the paper brochure is emailed to the user with a summary in the email body. This summary is also built by reasoning about the discourse tree. As in discourse-based approaches to summarisation, only the nuclei are kept leaving out all the satellites corresponding to content considered as optional.

SciFly follows a two-stage approach: during discourse planning, content and organisation are selected, and a discourse tree is built, as in (Moore and Paris, 1993). The discourse plans specify the RST that hold between text spans. They were written for SciFly, based on an analysis of a corpus of sample human-authored brochures. In the second stage, the presentation stage, the system reasons about the discourse tree and the characteristics of the delivery channel to decide how much content to include and how to present it.

The following section presents two algorithms that reason about how much to express based on the space available. The first one is fairly naïve, implementing the basic notions of how RST can help with this reasoning. The second algorithm addresses limitations discovered when we deployed the system as an information kiosk at a major IT fair.

4 Determining how much content to realise

During the discourse planning stage, SciFly retrieves all the information corresponding to the user's topics of interest and organises it into a coherent whole. As the output is to be a brochure-like document, presenting the work carried out in our organisation (e.g., projects and capabilities), the system includes more information than is strictly required. For example, an introduction about the organisation and relevant contact details are always

[3] It is worth noting that it took one person in the marketing team just a few days to write the content.

included, and, in the case of a brochure about a project, a description of the research laboratory in which the project resides is also provided.

At the end of this stage, the system has produced a discourse tree. It includes the top level communicative goal, the intermediate goals and the rhetorical relations that exist between text spans. It thus encodes both the purpose of each retrieved text fragment and how these fragments relate to each other. This provides a basis to reason about which information to realise when there is too much content for some delivery medium (e.g., a double-sided A4 page).

As noted earlier, our two algorithms embody the principle of nuclearity. They also both exploit the notion that some relations are more important than others. For example, *context* (providing the context in which to understand the information in the nucleus) might be considered more important than *elaboration* (providing more details). By assigning an importance value to relations, it becomes possible to rank the relations based on their contribution to the communicative goal. Our assignment, presented in Table 1, is based on judgments from our marketing staff.[4] The importance of each relation is defined in a declarative configuration file that can be easily modified to suit different preferences and application contexts.

Shading	Discourse Relations	Importance
Black	Illustration, Background, Circumstance, Elaboration	Low Low-Medium
Dark Grey	Context, Motivation, Evidence, Summary , Justification,	Medium
Light Grey	Preparation, Enablement	Medium-High High

Table 1: Some discourse relations and their ranking

To explain our algorithms, we represent the discourse tree using an abstract view, as shown in Figure 2. The communicative goals are represented as nodes. A white node indicates a nucleus; the other nodes, the satellites, are all shaded in grey corresponding to the importance of the rhetorical relation linking them to the nucleus.

Each node is the root of a subtree (empty if the node is a leaf) which generates some content. In both algorithms, the system computes for each node the approximate space required for that content in number of lines. This is computed bottom-up in an iterative manner by looking at the retrieved content at each node. Note, however, that the system can only compute an approximation of the number of lines of content to be generated, as this depends on style, line-wrapping and other formatting attributes within the text fragments and global spacing decisions in the PDF rendering process. In Figure 2, the number inside each node indicates the approximate amount of content that node produces (in lines).

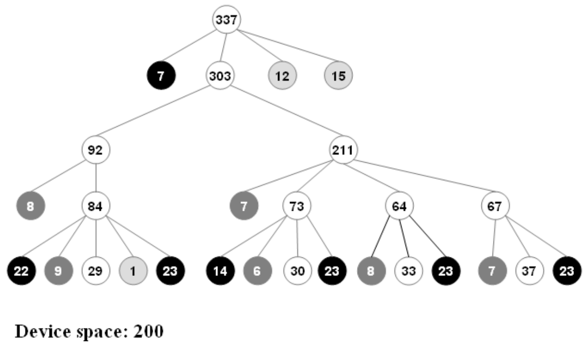


Figure 2: Discourse tree annotated with how much content would be generated for each node.

4.1 Naïve algorithm

From the two features of RST mentioned above (nuclearity and relative importance of relations), we designed a straighforward algorithm to exploit the discourse structure in order to decide how much content to realise. This is our "naïve algorithm". With this algorithm, the system examines the top level node to determine if the current structure will result in too much content, given the properties of the output medium (e.g., lines of content per page). If so, the system selects the relation with the lowest importance and traverses the tree, dropping all the satellite nodes (including their sub-trees) that are related to a nucleus with this relation. The algorithm repeats this process until the amount of content meets the device space constraint.

Consider the example in Figure 2; the top node indicates 337 lines of content, while the space available for the device is 200 lines (specified in a device model). The *illustration* relation is the least important relation present. The algorithms thus drops all the satellites related by *illustration*. Since this is not enough, it picks the next least important relation (*background* in this case) and repeats the process until the space requirements are met. The resulting discourse tree and the ordered list of dropped nodes are shown in Figure 3. It is important to note that, with this approach, the resulting document is shorter but still coherent. This is because reasoning is done with the discourse tree and the rhetorical relations holding between subgoals.

We deployed the system with this algorithm at a trade fair in 2005 and 2006 and measured the general experience visitors had with the system. On average, people rated the system positively,

[4] SciFly actually has 5 levels of importance. We have merged "low" with "low-medium" and "high" with "medium-high" here to avoid too many shades of grey.

emphasising the nice layout of the brochure, its conciseness and its informative content. The area identified as needing improvement was the amount of blank space in brochures (as seen in Figure 1), where it seems that more information could have

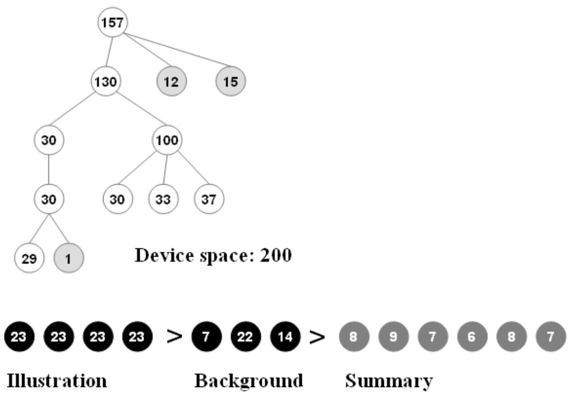


Figure 3: The resulting discourse tree after applying the naïve space constraint algorithm.

been included. This is because our naïve algorithm drops many sub-tree(s) at once, thus potentially deleting a lot of content at each step. For example, in its third pass, the algorithm deleted 45 lines in one go. This can be too much, and indeed, the resulting brochure can appear odd because of the excessive blank space. This led us to our enhanced algorithm.

4.2 The enhanced algorithm

We redesigned the algorithm to gain finer control over the text length. To do so, we take into account the depth of a node in addition to its rhetorical status. We first converted the *importance value* of a rhetorical relation (e.g., low, medium, high) into a *penalty score*. In our system, penalty scores range from 1 to 6, in increments of 1: from high importance relations with a score of 2, through to low importance relations with a score of 6. A nucleus is given a score of 1 to take the increment in tree depth into account.

We then assign each node an importance rating called its *weight*. It is computed by adding (1) the weight of the node's parent, to take into account the depth of the node in the whole tree, and (2) the *penalty score* of the rhetorical relation which relates the node to its nucleus. A child node is thus *heavier* than its parent. The larger the weight, the less important the node is to the overall discourse structure.[5]

Once the weights have been computed, the system orders the nodes by their weight, and the heaviest nodes get dropped first. Thus, nodes deeper in the tree and linked by a discourse relation with a high penalty score (low importance) get removed first. Nodes are now dropped one by one until the top level node has an amount of content that satisfies the space requirement. This provides finer control over the amount of realised content and avoids the limitation of the first algorithm.

We illustrate this process through the same example. We annotate the tree of Figure 2 with node weights, as shown in Figure 4 (the weights appear outside the nodes). The system can now order the satellite nodes, from *heaviest* to *lightest*. As we consider that nuclei are important, other nodes (i.e., satellites) are dropped preferentially at any given depth of the tree. This is why we do not include the nuclei in our ordered list (shown at the bottom of Figure 4). A nucleus will get dropped *only* if its parent is dropped. The system takes the nodes with the heaviest weight and drops them one by one, until the top node has the appropriate amount of content. In our example, the seven left most nodes of the ordered list will be dropped (as indicated in the Figure in the boxed area). This results in much less text being dropped than with the naïve algorithm to satisfy the same requirement (e.g., 137 lines dropped instead of 180). As before, pruning the tree does not affect the coherence of the resulting document.

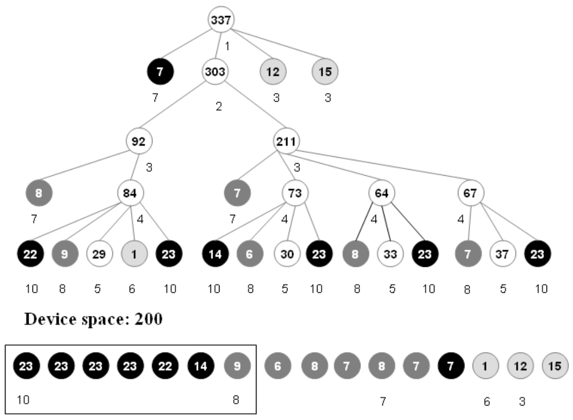


Figure 4: Tree annotated with weights and ordered list of (satellite) nodes

As mentioned previously, techniques used in discourse-based approaches to summarisation are quite similar to our algorithm. Like we do, they take advantage of the discourse structure and exploit the difference between nuclei and satellites to determine the most important units in a text. However, there are a number of differences that influence the ordering based on the importance of the nodes (units of text) and how much gets deleted. For example, the granularity of the units of text is different. This might have an impact on the type of discourse representation built (e.g., our satellites nodes

[5] Note that this is similar in concept to the relevance rating proposed by O'Donnell (1997), but computed differently. O'Donnell's rating penalises only nodes in a satellite scope, thus only partially taking the depth of the tree into account.

are rarely further decomposed). Moreover, although Marcu (1998) has suggested that the semantics of rhetorical relations may play a major role in determining the important units in a text, this was not integrated in his scoring function because of lack of empirical evidence in his data. We believe that these semantics not only influence the ordering of the nodes but also lead to a finer grained partial ordering, thus giving us finer grained control.

To illustrate this, we applied Marcu's scoring function to the discourse tree of Figure 2. Figure 5 shows the partially ordered list of nodes we obtained (ranked from most to least important). While the two lists are not fundamentally different, the way nodes get dropped is different. With Marcu's scoring function, we would have dropped 159 lines instead of 137 using a partial ordering with 4 levels instead of the 6 with our enhanced algorithm. Finally, our treatment of lists and itemised items is substantially different, as described below.

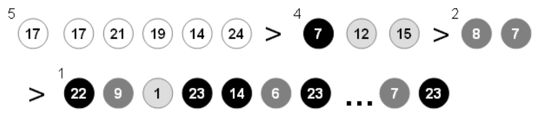


Figure 5: Ordered list of nodes when applying Marcu's scoring function

This enhancement to the algorithm allows the system to have finer control over what is generated. However, as noted earlier, shortening a text, rather than summarising it, raises a new challenge: that of ensuring that the final text is still balanced. Sometimes, a discourse structure contains several parallel sub-structures that, if pruned unevenly, results in text that is unbalanced and appears odd. This happens for example in a paper, when we itemise or enumerate to provide a list of topics. We typically try to balance the amount of text each topic contains, avoiding having one topic with one sentence while another topic has several paragraphs. In our case, we have such parallel structures when describing a list of projects (belonging to a research laboratory, for example, or when the user has indicated interest in several projects). In these cases, the discourse structure contains several sub-trees which are likely to be of the same structure, as illustrated schematically in Table 2. This structure is generated during discourse planning by a plan containing a *foreach* statement, e.g., *(foreach project in project-list (describe project)).*

Project 1	**Project 2**	**Project 3**
• Introduction	• Introduction	• Introduction
• Description	• Description	• Description
• Illustration	• Illustration	• Illustration

Table 2. Example of parallel and balanced structures

To keep the overall structure balanced, the system annotates all sub-structures issued from such a foreach statement. Then, when constructing the ordered list of satellites, the system clusters nodes at the same level of depth in the sub-structures, taking into account their relationship to the nucleus, as shown in Figure 6. (Note that, since these sub-trees are generally identical, the nodes will often have similar weights.) When dropping nodes, the whole cluster will be deleted at the same time, rather than node by node.

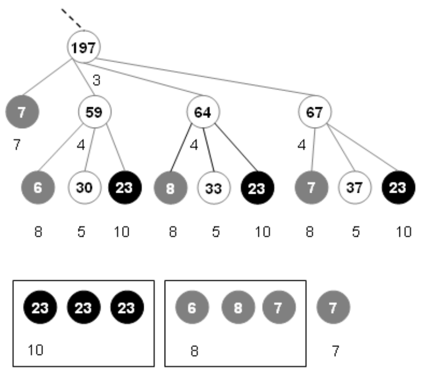


Figure 6: Ordered list of (satellite) cluster nodes

In the structure of Table 2, illustrated in Figure 6, for example, illustrations for all projects (cluster of weight 10) will be dropped together, then all introduction sections (cluster of weight 8). This prevents one sub-structure from being pruned more than its sibling structures and thus ensures the resulting brochure is balanced.

5 Comparative Performance Analysis

We performed an analysis of the two algorithms to assess their comparative effectiveness with respect to filling the space of a two-page brochure. In particular, we wanted to find out whether the enhanced algorithm reduced the amount of dropped content, filling up the available space more effectively.

We automatically generated 1605 brochures about randomly selected topics, using both algorithms. For each brochure, we stored data about the total amount of content initially assembled and the amount of content withheld from the generated brochure (none if nothing was dropped). For our analysis, we kept only the brochures for which content was withheld. This left us with 1507 brochures. We observed the following improvements, as shown in Figure 7:

- 82.5% of the brochures generated with the enhanced algorithm filled over 96% of the available space (leaving at most 8 lines of empty space). Only 29% of brochures generated with the naïve algorithm achieved this performance.

- 96.5% of the brochures generated with the enhanced algorithm filled at least 90% of the space, compared with 44.5% of brochures generated using the naïve algorithm.

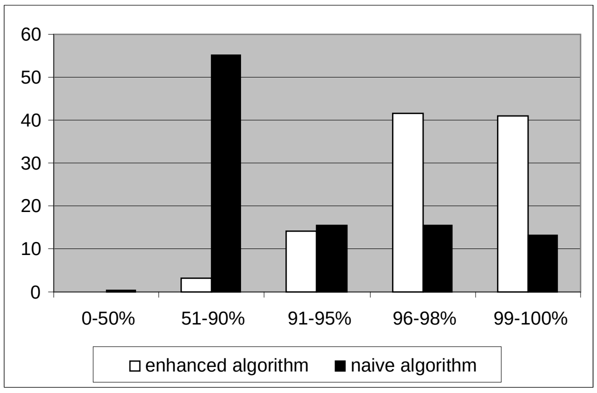


Figure 7: Percentage of fullness of the brochures (vertical axis is the number of brochures, in percentage)

Given the discourse planning approach employed by our system, we know that the generated text is coherent with both algorithms. The results on the amount of text generated show that our enhanced algorithm results in a much better use of the available space than the naïve algorithm. Finally, because the algorithm prunes parallel structures in a synchronised manner, we know that the resulting text is balanced. We have thus achieved our goal of producing a well structured coherent text that fills as much of the space available as possible, when fine-grained control over the text generated is not possible. We can also conclude that it is useful to exploit the discourse structure to reason about what to include when we have specific space requirements.

In further analysing the results, we found the following:

- 75% of brochures included more content using the enhanced algorithm , as we desired;
- 13% had the same amount of content regardless of the algorithm used. (Note that the same amount of content does not necessarily mean that the content is identical, as the algorithms select the content to drop differently.); and
- 12% of the brochures actually contained less content with the enhanced algorithm than with the naïve one.

We examined these results in more detail to understand what was happening.

In the brochures that gained content with the new algorithm, an average of 32 new lines of content was included, which represents about 15% of the whole brochure. More specifically, as shown in Figure 8: 36% of the brochures gained between 1 and 20 lines of relevant content (representing 1-10% of the brochure); 28% gained between 21 and 40 lines (11-20% of the brochure); and 30% gained 41 to 60 lines (21-30% of the brochure).

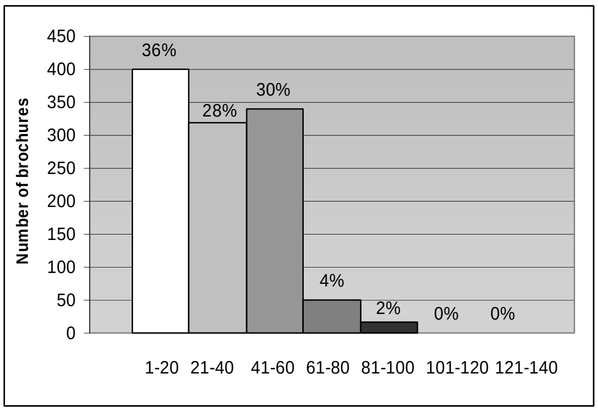
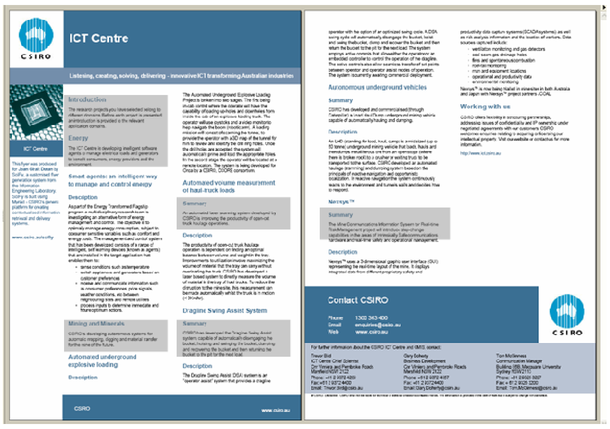


Figure 8: Percentage of gained lines with the enhanced algorithm

An example of brochure generated with the enhanced algorithm is shown in Figure 9. The content kept by the enhanced algorithm that would have been dropped by the naïve one is highlighted in grey.



ICT Centre

Listening, creating, solving, delivering - innovative ICT transforming Australian industries

Contact CSIRO

Figure 9: Example showing the differences in content between the two algorithms

In the 12% of cases where the enhanced algorithm drops more content than the naïve one, we noted that, on average, the naïve algorithm had produced a brochure which was about 3 lines away from a full brochure, while the enhanced one produced a brochure which was, on average, 7 lines away from such a brochure. Thus, these brochures were at least 96% filled for both algorithms. The reduction in realised content for the new algorithm was due to our treatment of parallel discourse structures, thus representing a desirable loss of content to create balanced brochures, as described earlier. This illustrates a limitation of this metric for comparative performance analysis: it only takes into account the space used rather than the overall quality of the output. Clearly loss of content is desirable in some cases to maintain a well structured and balanced text. To measure the overall quality of the brochures produced by the two algorithms, we performed a user evaluation.

6 User evaluation

In our user evaluation, we asked users to compare pairs of brochures, where one brochure was produced by the naïve algorithm and the other by the enhanced algorithm. The users were asked to choose which brochure, if any, they preferred. The aim was to ensure that the improved use of space did not have any negative impact on the overall quality of the brochure and the users' satisfaction. The layout for all brochures was kept constant.

Seventeen users participated in the evaluation and were presented with seven pairs of brochures. The pairs of brochures were selected to represent the variety of output produced by both algorithms. As mentioned earlier, in the majority of the cases, the enhanced algorithm uses the available space more effectively by including more content in the brochure. However, in a number of cases, due to our treatment of parallel structures, and also because the two algorithms select the content to drop differently, the enhanced algorithm produces brochures with the same amount or less content than using the naïve one. To represent these cases and evaluate whether this has an impact on how users assess the quality of the brochures, we selected the brochures as follows: in three pairs, the brochures generated by the enhanced algorithm contained more content (cluster 1), in two pairs, both brochures had the same amount of content regardless of the algorithm used (cluster 2), and in two pairs, the brochures generated by the enhanced algorithm contained a bit less content (cluster 3). To control any order effect, the pairs were randomly presented from user to user, and in each pair, each brochure was randomly assigned a left-right configuration. The results are shown in Table 3.

Pairs	**ENHANCED**	**NAIVE**	**EQUIV**
Cluster 1	26	9	11
Cluster 2	8	14	12
Cluster 3	17	9	8
Total	**51**	**37**	**31**

Table 3: Users' ratings of preference

Table 3 shows that participants generally preferred the brochures generated with the enhanced algorithm over the ones produced with the naive algorithm, or found them equivalent. If we group together the participants' preference for the brochures generated by the enhanced algorithm (51 votes) with the 31 cases where participants found the pair of brochures equivalent, we see that we have not lost any performance in terms of users' satisfaction with the enhanced algorithm, while we gain significantly in the amount of text included.

Interestingly, most users preferred the enhanced algorithm for cluster 3, where the enhanced algorithm produced less text but pruned parallel substructures in an even manner, resulting in more balanced documents. Also, users seem to prefer the naïve algorithm for cluster 2, when both algorithms produce the same amount of text. After further analysis, we found that users liked having text presented as *summary*, a relation which got dropped more often with the new algorithm. This can be addressed by changing the importance of this relation. We also asked the users to explain their choice. 'Good presentation' and 'good flow of information' were the reasons given for preferring the brochures generated with the enhanced algorithm.

7 Conclusions

The ability to reason about how much content to generate in order to convey a message under space constraints is an important consideration for NLG systems. In our applications, we cannot resort to the traditional method of controlling the lexical-grammatical resources to that effect. We generate text by re-using existing text fragments over which we do not have any control, and we need to produce, at discourse planning stage, a discourse tree with all the appropriate available content, in order to realise the output on several delivery channels (e.g., full structure for a web output, subset of the structure to fit specific space constraints such as a double-sided A4 page and only nucleus content for the email summary). We thus had to find other ways to satisfy space requirements. To this end, we implemented two algorithms that reason about the discourse tree. The first naïve algorithm, embodying in a straighforward manner the notions of nuclearity and the rhetorical relations' relative importance, resulted in a sub-optimal use of space. The enhanced algorithm addressed this limitation and ensured a balanced text. Our comparative analysis showed that our enhanced algorithm produces documents filling most of the available space, while maintaining users' satisfaction.

Acknowledgements

We would like to thank Meriem Raji for her work on the user experiment, all the people who took part in our evaluation, our group and Keith Vander Linden for their input throughout the project.

References

John Bateman, Thomas Kamps, Jörg Kleinz and Klaus Reichenberger (2001). Constructive text, diagram and layout generation for information presentation: the DArt_bio system. *Computational Linguistics*, 27(3):409-449.

Nathalie Colineau, Cécile Paris and Mingfang Wu. (2004). Actionable Information Delivery. *Revue d'Intelligence Artificielle (RSTI – RIA)*, Special Issue on Tailored Information Delivery, 18(4), 549-576.

William C. Mann and Sandra A. Thompson. (1988). Rhetorical Structure Theory: Toward a functional theory of text organisation. *Text* 8(3):243-281.

Daniel Marcu. (1998). To build text summaries of high quality, nuclearity is not sufficient. In *Working Notes of the AAAI-98 Spring Symposium on Intelli-gent Text Summarization*, Stanford, CA, 1–8.

Johanna D. Moore and William R. Swartout. (1989). A Reactive Approach to Explanation. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence (IJCAI 1989)*, 1504-1510.

Johanna D. Moore and Cécile L. Paris. (1993). Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information, *Computational Linguistics*, 19 (4):651-694, Cambridge, MA.

Kenji Ono, Kazuo Sumita and Seiji Miike (1994). Abstract generation based on rhetorical structure extraction. In *Proceedings of the 15th Conference on Computational Linguistics* (CoLing 1994), Volume 1, Kyoto, Japan, 344 – 348.

Mick O'Donnell (1997). Variable Length On-Line Document Presentation. *Proceedings of the 6th European Workshop on Natural Language Generation*. Gerhard-Mercator University, Duisburg, Germany.

Cécile Paris, Nathalie Colineau, Keith Vander Linden and Andrew Lampert (2008). Myriad: A Reusable Platform for Tailored Information Delivery. CSIRO Technical Report 08/050.

Ehud Reiter. (2000). Pipelines and Size Constraints. *Computational Linguistics*, 26:251-259.

Lucia H.M. Rino and Donia R. Scott. (1994). Automatic generation of draft summaries: heuristics for content selection. In *Proceedings of the 3rd International Conference on the Cognitive Science of Natural Language Processing*, Dublin.

Donia R. Scott and Clarisse S. de Souza. (1990). Getting the message across in RST-based text generation. In Dale, Mellish & Zock (eds). *Current Resarch in Natural Language Generation.* London: Academic Press. 119-128.

Karen Spark Jones (1993). What might be in a summary? *Information Retrieval 93:* Von der Modellierung zur Anwendung. (Ed: Knorz, Krause and Womse-Hacker), Konstanz: Universitatsverlag Konstanz, 0-26.

Maite Taboada and William C. Mann. (2006). Applications of Rhetorical Structure Theory. *Discourse Studies*, 8(4):567-588.

Keith Vander Linden. (2008). A Dynamic Programming Approach to Document Length Constraints. In *Proceedings of the Fifth International Conference on Natural Language Generation*, June 12-14, 2008, 177-180.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 142-150

A Two-Level Morphological Analyser for the Indonesian Language

Femphy Pisceldo, Rahmad Mahendra & Ruli Manurung
Faculty of Computer Science
University of Indonesia
fep40@ui.edu,rama42@ui.edu,maruli@cs.ui.ac.id

I Wayan Arka
Department of Linguistics
Australian National University
wayan.arka@anu.edu.au

Abstract

This paper presents our efforts at developing an Indonesian morphological analyser that provides a detailed analysis of the rich affixation process[1]. We model Indonesian morphology using a two-level morphology approach, decomposing the process into a set of morphotactic and morphophonemic rules. These rules are modelled as a network of finite state transducers and implemented using **xfst** and **lexc**. Our approach is able to handle reduplication, a non-concatenative morphological process.

1 Introduction

Morphology is the study of the way that words are built up from smaller units called morphemes, the minimal meaning-bearing units in a language (Jurafsky, 2000). For example, the English word *kind* consists of a single morpheme (the root word *kind*) whilst the word *players* consists of three morphemes: *play*, *-er* and *-s*. The morphemes *kind* and *play* can stand alone as words, while affixes *-er* and *-s* must appear bound to another morpheme.

By applying a set of morphological rules, we can produce not just morphemes but also other information relating to the words; for example, the grammatical category of the whole word as well as the subcategorisation frame of the word if it is a verb. This process is called morphological analysis. In this respect, the value of a morphological analyser would be twofold: from a theoretical (linguistic) viewpoint, it is a very useful tool for linguistic modelling and for testing certain analyses. On the other hand, from a practical viewpoint, it supports many applications, e.g. information retrieval, search engines, and machine translation, among others.

There is currently some interest in developing morphological tools for the Indonesian language. In previous work, Siregar (1995) and Adriani et al. (2007) discuss the development of Indonesian stemmers that recover a root from an affixed word, implemented procedurally. However, stemmers are of limited use as they do not provide any more linguistic information beyond the stem. Hartono (2002) presents an initial version of a morphological analyser developed with PC-KIMMO, but unfortunately, it does not handle reduplication, a key aspect of Indonesian morphology. The morphological analyser that we are developing and that we describe here is designed to be able to handle the rich semantic, lexical and grammatical information associated with words and word formation in NLP applications.

In Section 2, we first discuss Indonesian morphology, followed by a brief explanation of two-level morphology in Section 3. Sections 4 and 5 present our work in applying two-level morphology for the Indonesian language. Finally, Section 6 presents the results of some evaluations we carried out on our developed analyser.

2 Indonesian Language Morphology

Indonesian is a variety of Malay, which belongs to the Austronesian language family (Gordon, 2005; Sneddon, 2003). It is spoken by about 190 million people in Indonesia and other parts of the world[2].

[1] This research is part of a collaborative research project funded by ARC Discovery Grant DP0877595.

[2] According to *Biro Pusat Statistik*, as of 2004.

Words in Indonesian are built from their roots by means of a variety of morphological operations including compounding, affixation, and reduplication. Indonesian concatenative morphology regulates how a stem and its affixes glue together, while a non-concatenative one combines morphemes in more complex ways.

Affixes in Indonesian can be classified as four categories (Alwi et al., 2003). Prefixes precede the base form, i.e. *meN-*, *di-*, *peN-*, *per-*, *ke-*, and *ter-*. Suffixes follow the base form, i.e. *-kan*, *-an*, and *-i*. Infixes are inside the base form, i.e. *-el-*, *-em-*, and *-er-*. Circumfixes wrap around the base form. While circumfixes formally are combinations of allowed prefixes and suffixes, they have to be treated as discontinuous units for semantic and grammatical reasons. In our Indonesian morphological analyser, we have handled prefixes, suffixes, and circumfixes.

Indonesian non-concatenative morphology refers to reduplicated morpheme forms. Reduplicated words based on morpheme regularity are grouped into full reduplication (e.g., the word *buku-buku* is derived from the stem *buku*) and partial reduplication of different kinds. The latter includes reduplicated stems with affixes (e.g. the word *buah-buahan* is derived from stem *buah*, *bertingkat-tingkat* is derived from stem *bertingkat*) and various (often rather irregular) reduplications (e.g. *sayur-mayur* is derived from *sayur*).

Indonesian affixation and reduplication are illustrated in the following example. From the stem *pukul*, we can derive words like *pemukul* (by concatenating the prefix *peN-*), *memukuli* (by concatenating the circumfix *meN-i*), and *pukulan-pukulan* (by first concatenating the suffix *–an*, then applying full reduplication). Other examples include *dipukul, pemukulan*, and *berpukul-pukulan*.

All kinds of regular word-formation with or without reduplication are recognized in this research.

Morphology generally makes a distinction between inflectional and derivational processes. In the previous example, the formation of *memukuli* appears to be ‘inflectional’ as the formation does not change the category of the stem, and the formation of *pemukul* and *pukulan-pukulan* is derivational because the derived words are nouns while the stem *pukul* is a verb. However, the distinction between inflection and derivation, particularly in Indonesian, is not always clear cut. The formation of verbs such as *memukuli* from *pukul* is arguably derivational in nature because the new words have quite different lexical properties, even though both the new verbs and the stems are of the same category (i.e. ‘verb’).

3 Two-Level Morphology

A morphological analyser can be used to process a list of morphemes in a lexicon to yield fully derived words. In the other direction, it can be used to identify affixes and roots from derived words. For example, taking the word *membaca* as input, an Indonesian morphological analyser should produce the string *baca*`+Verb+AV`, indicating that the active verb *membaca* is derived from *baca*.

However, it is difficult to accomplish morphological analysis in one step only. In Indonesian, affixation does not consist of just fusing the affix with the stem. Modification of phonemes may be necessary, e.g. the concatenation of *meN+putar* will produce word *memutar*, while the result of *meN+siram* is *menyiram*.

To solve this problem, we adopt a two-level morphology model. Two-level morphology is based on finite-state transducers (Koskenniemi, 1983). In practice, this model has been successfully applied to several languages such as English, Finnish, Japanese, Russian, and French. It is not only useful to account for various concatenative morphologies, but is also able to handle non-concatenative processes, as has been developed in Malagasy tokenization and morphological analysis (Dalrymple et al., 2006).

4 Design

Our design for an Indonesian morphological analyser is divided into two components: **morphotactic** rules that explain which classes of morphemes can follow other classes of morphemes inside a word, and **morphophonemic** rules which model the changes that occur in a word. The rules in each component are typically applied in parallel. Additionally, these rules are combined with a lexicon of stems in order to complete the full design.

A word, in order to be analysed, will follow the path lexicon → morphotactic rules → morphophonemic rules→ surface. Before the result of the morphological analyser appears at the surface, it will follow the lexicon path to determine the actual morpheme of that word. After moving from the lexicon, that word will be analysed by morphotactic and morphophonemic rules. Only after finishing the process in morphotactic and morphophonemic rules, will the result of morphological analyser for that word be delivered.

The explanation below will explore, in a little more depth, the design of lexicon, morphotactic rules, and morphophonemic rules.

4.1 Lexicon design

Our lexicon equates to a set of Indonesian stem words. Affixes are not stored in the lexicon as they are already accounted for by the morphotactic rules.

For our initial design, our lexicon is divided into four classes, i.e. verbs, nouns, adjectives, and 'etc', which includes all other stems, e.g. pronouns, adverbs, numbers, and particles. Clustering these word classes together is certainly a large oversimplification, and one which we hope to address in future revisions.

4.2 Tag design

The design of tags has become very important in the development of morphological analysers, since the tags will deliver linguistic information that occurs on a word being analysed.

In our research, the tags to be designed can be divided into normal tags and special tags. Normal tags can be output by the morphotactics component without any circumstances, whilst special tags only occur if the involved stems are associated with specific markers in the lexicon. The normal tags are **+VERB**, **+NOUN**, **+ADJ**, **+BAREVERB**, **+BARENOUN**, **+BAREADJ**, **+BAREETC**, **+AV**, **+PASS**, **+UV**, and **+REDUP**, whilst the special tags are **+CAUS_KAN**, **+APPL_KAN**, **+CAUS_I**, **+APPL_I**, **+ACTOR**, **+INSTRUMENT**.

Tags like **+VERB**, **+NOUN**, and **+ADJ** give the part of speech (POS) information for fully inflected words, whereas **+BAREVERB**, **+BARENOUN**, **+BAREADJ**, and **+BAREETC** give the POS information for stems.

Other tags provide important linguistic information, such as voice, a grammatical category playing a central role in Indonesian syntax, e.g. **+AV**, which indicates active voice, **+PASS**, which indicates passive voice, and **+UV**, which is reserved for undergoer voice.

Furthermore, **+REDUP** is a tag indicating reduplication. **+CAUS_KAN**, **+CAUS_I**, **+APPL_KAN**, and **+APPL_I** show whether words are causative or applicative with respect to their suffixes. The last two tags (**+ACTOR** and **+INSTRUMENT**) indicate either it is an actor or an instrument that is being brought with the construction of that word.

4.3 Morphotactic rules

In designing a morphological analyser, morphotactic rules are crucial to model how two or more morphemes can be merged.

Based on (Alwi et al., 2003), the morphotactic rules for Indonesian can be classified into 13 classes. Ten of these classes are determined based on which suffixes are merged with the stem, while the other three are reduplication cases. The first ten classes can be identified as concatenative morphology, while the other three are nonconcatenative morphology.

In our design, the ten classes that belong to concatenative morphology are the morphotactic rules for the prefixes *meN*-, *peN*-, *di*-, *per*-, *ber*-, *ter*-, and *ke*-, and the morphotactic rules for the suffixes -*an*, -*kan*, and -*i*. Some example cases that belong to these 10 classes are as follows:

- *meN* + stem(verb, noun, adjective or etc) + *kan* → Verb. Example: *membersihkan* (*meN*+*bersih*+*kan*). In this example, the word *bersih* is an adjective. After merging with *meN*-*kan*, the resulting word is a verb.
- *peN* + *ber* + stem(verb, noun, adjective or etc) + *an* → Noun. Example: *pembelajaran* (*peN*+*ber*+*ajar*+*an*). In this example, the word *ajar* is a verb. After merging with *peN*-*ber*-*an*, the resulting word is a noun.
- *ke* + *ber* + stem(verb, noun, adjective or etc) + *an* → Noun. Example: *keberhasilan* (*ke*+*ber*+*hasil*+*an*). In this example, the word *hasil* is a noun. After merging with *ke*-*ber*-*an*, the resulting word is also a noun.
- stem(verb, noun or adjective) + *i* → Verb. Example: *terangi* (*terang*+*i*). In this example, the word *terang* is an adjective. After merging with –*i*, the resulting word is a verb.

The last three classes, which belong to the category of nonconcatenative morphology, are the morphotactic rules for full reduplication, partial reduplication, and affixed reduplication. Some example cases that belong to these three classes are as follows:

- Full reduplication without affixation. Example: *buku-buku*. In this example, the word *buku-buku*, which is a noun, is generated from the word *buku*, which is also a noun.
- Full reduplication with *ke-an*. This case has the following rule: reduplication of (*ke*+stem (bare verb, adjective or etc)+*an*) → Noun. Example: *kekayaan-kekayaan* (reduplication of *ke*+*kaya*+*an*). In this example, the word *kaya* is an adjective. After the morphological process, the resulting word is a noun.
- Partial reduplication with *ber-*. This case has the following rule: *ber*+ reduplicated stem (bare verb) → Verb. Example: *berlari-lari* (*ber*+*lari-lari*). In this example, the word *lari* is a verb. After the morphological process, the resulting word is also a verb.
- Affixed reduplication with *meN-*. This case has the following rule: stem (bare verb) +*meN*+stem (bare verb) → Verb. Example: *tanam-menanam* (*tanam-meN+tanam*). In this example, the word *tanam* is a verb. After the process, the resulting word is also a verb.

During the stage of morphotactic rules, there are several steps that must be followed in order to complete the process. Those steps include the addition of prefixes and preprefixes, the addition of stems and part-of-speech, the addition of suffixes, and the final processing of additional tags. After finishing all of these steps, the process moves on to the morphophonemic process.

4.4 Morphophonemic rules

All the rules that define how two or more morphemes can merge have been designed in morphotactic rules. However, the merging process is still not completed; hence we still have to define what changes have to be made after these morphemes merge. For these issues, we define morphophonemic rules that define the phonetic changes that occur.

In Indonesian, these rules can generally be divided into two groups. The first group consists of four rules that model the phonetic changes in stems, whereas the second group consists of seven rules that model the phonetic changes in affixes.

Based on (Alwi et al., 2003), the four rules in the first group are:

- */k/* replacement with */ng/* if the word starts with *meN-* or *peN-*. Example: *meN+kantuk* → *mengantuk*.
- */s/* replacement with */ny/* if the word starts with *meN-* or *peN-*. Example: *peN+sebaran* → *penyebaran*.
- */p/* replacement with */m/* if the word starts with *meN-* or *peN-*. Example: *peN+pakai* → *pemakai*.
- */t/* replacement with */n/* if the word starts with *meN-* or *peN-*. Example: *meN+tertawakan* → *menertawakan*.

Note that the rules above only change the stem-initial segment. To complete the replacement processes, the following seven rules from the second group are needed to get the correct prefix forms:

- */N/* deletion if *meN-* is followed by */l/, /m/, /n/, /r/, /y/, /w/, /t/, /s/, /p/, /k/* or if there is *peN-* followed by */l/, /m/, /n/, /r/, /d/, /w/, /t/, /s/, /p/, /k/*. Example: *meN+lukis* → *melukis*.
- */r/* deletion if there is *ber-*, *ter-*, or *per-* followed by */r/* or the word that its first syllable ended with */er/*. Example: *ber+runding* → *berunding*.
- */N/* replacement with */n/* if there is *meN-* followed by */d/, /c/, /j/, /sy/* or there is *peN-* followed by */d/, /c/, /j/*. Example: *peN+jual* → *penjual*.
- */N/* replacement with */m/* if there is *meN-* or *peN-* followed by */b/, /f/*. Example: *peN+buru* → *pemburu*.
- */N/* replacement with */nge/* if there is *meN-* followed by a word that has only one syllable. Example: *meN+rem* → *mengerem*.
- */N/* replacement with */l/* if there is *peN-* followed by the word *ajar*. Example: *peN+ajar* → *pelajar*.
- */r/* replacement with */l/* if there is *ber-* followed by the word *ajar*. Example: *ber+ajar* → *belajar*.

After all sub-processes invoked by the rules in the first group and the second group are parallelized, then the whole morphophonemic process is finished.

The design of morphophonemic rules for reduplication is very much the same as the one in affixation, since basically the morphophonemic process in reduplication occurs in the affixation part of reduplication.

However, several rules, in particular those that model morphophonemic processes where both affix and stem undergo changes, had to be revised to account for their behaviour when applied to reduplicated forms. For example, the morphophonemic rules '*/k/* replacement with */ng/*' and '*/N/* deletion', which work in tandem, were originally defined as '*/k/* deletion' and '*/N/* replacement with */ng/*'. However, this approach would not have worked for cases involving reduplication. For example, the word *mengotak-ngotakkan* will not be analysed properly if the morphological analyser uses the rules '*/k/* deletion' and '*/N/* replacement with */ng/*'. The word *mengotak-ngotakkan* is generated from the stem *kotak* that is modified with affixed reduplication *meN-kan*. If '*/k/* deletion' and '*/N/* replacement with */ng/*' are being used, the word will become *mengotak-otakkan* which is not valid. This is why, for the sake of reduplication, the rule is changed to '*/k/* replacement with */ng/*' that is parallelized with '*/N/* deletion'. Consequently, *mengotak-ngotakkan* can be properly generated.

5 Implementation

This Indonesian morphological analyser is implemented in **`xfst`** and **`lexc`** (Beesley & Karttunen, 2003). The morphotactic rules are implemented in **`xfst`** while morphophonemic rules are implemented in **`lexc`**.

5.1 Morphotactic rules implementation

The morphotactic rules can be illustrated as the finite-state automata shown in Figure 1. Valid Indonesian words, i.e. those that are constructed through legal morphological processes, are accepted by the automata, whereas invalid words are rejected.

Starting from the root, each state defines the next possible state whilst emitting (or consuming) a certain symbol. In **`lexc`**, these states are called continuation classes. All continuation classes reachable from the root represent prefixes and *pre*-prefixes. The distinction between the two is necessary to encode the possible morphological variations that take two prefixes, e.g. *memper-*, *diper-*. From there, the next continuation class is the **`Stem`**, where the root word is emitted or consumed. This is subsequently followed by several classes representing possible suffixes, but there are also **`Redup1`** and **`Redup2`** classes that appear before and after the suffixes. Their function is to handle reduplication (Section 5.3). Lastly, the **`TagEmit`** class processes

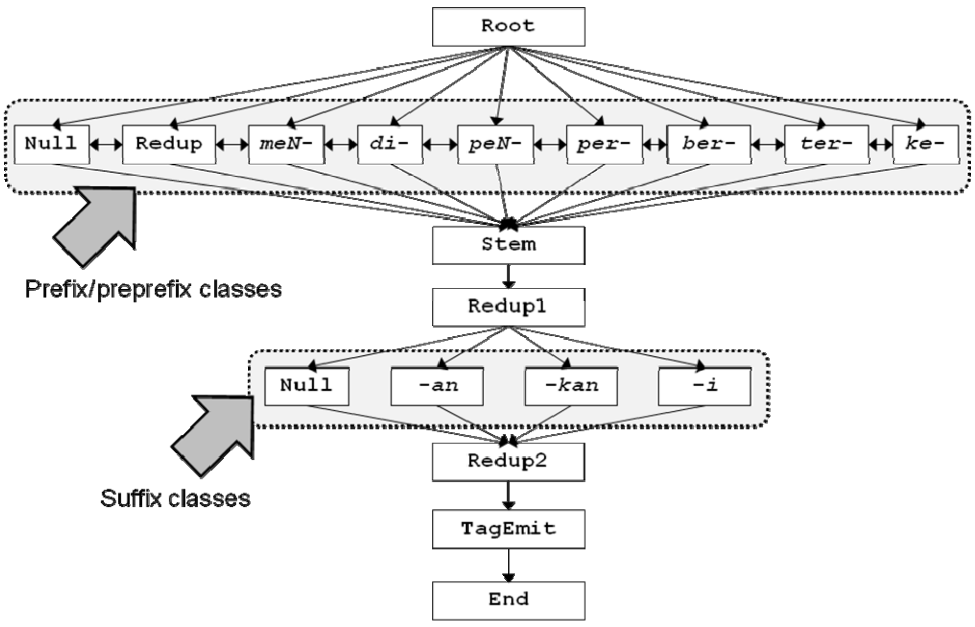


Figure 1. Illustration of Process Flow

all tags not yet handled by preceding classes (Section 4.2).

Throughout this morphotactic process, we extensively employ *flag diacritics*, a crucial feature of **`lexc`** which approximates the power of feature structures, i.e. being able to specify certain constraints to ensure that only valid paths of the network may be traversed. One benefit of this approach is the maintenance of a compact network representation. There are three flag diacritics used in our model: positive setting (**`@P.feat.val@`**), required test (**`@R.feat.val@`**), and disallow test (**`@D.feat.val@`**). Using these diacritics, we are able to stipulate values and constraints of certain aspects, which must be consistent throughout a path.

For example, the prefix *meN-* may combine with the suffix *–kan* and *–i*, but not *–an*. We can represent this with a feature, e.g. **`PREF`**, which is set to **`meN`** at the appropriate (pre)prefix state (**`@P.PREF.meN@`**). Suitable tests are then added to the suffix states, e.g. if the current path went through the *meN-* prefix state, **`@R.PREF.meN@`** stipulates that a suffix can be applied, whereas **`@D.PREF.meN@`** prevents a suffix from being applied.

To further exhibit the usage of flag diacritics in our morphological analyser, we provide an example of analysing the word *memukuli*, which consists of the prefix *meN-*, followed by the stem *pukul*, and the suffix *–i*. Figure 2 shows the path through the network used to analyse this word.

Starting from the root, the analyser parses the prefix *meN-* in *memukuli* and appropriately sets the flag diacritic for the feature **`PREF`**. This can be seen by the following **`lexc`** rule snippet:

```
LEXICON PrefixMeN
<[0 .x. m e "^N"] "@P.PREF.meN@"> Stems;
```

The **`LEXICON PrefixMeN`** line defines the state in the network that represents the meN- prefix. The next line defines a state transition where, by emitting or consuming the meN- prefix, the analyser can proceed to the **`Stems`** continuation class. Additionally, the **`PREF`** flag is set to **`meN`**. In reality, the **`PrefixMeN`** state has five other possible transitions, but they are not presented here.

Subsequently, at the **`Stems`** state, the analyser positively sets the flag diacritic for the feature **`STEM`** with value **`verb`**. Various required and disallow test flags will be called at the **`Redup1`**, **`SuffixI`**, **`Redup2`**, and **`TagEmit`** stages. However, for now, we focus on the **`SuffixI`** state, detailed in the following **`lexc`** rule **`snippet`**:

```
LEXICON SuffixI
<"+Verb":i "+AV":0 "@D.PREPREF@" "@D.REDUP@"
["@R.STEM.Vrb@"|"@R.STEM.Nom@"|"@R.STEM.Adj@"]
"@R.PREF.meN@" "@P.SUFF.i@"> Redup2;
```

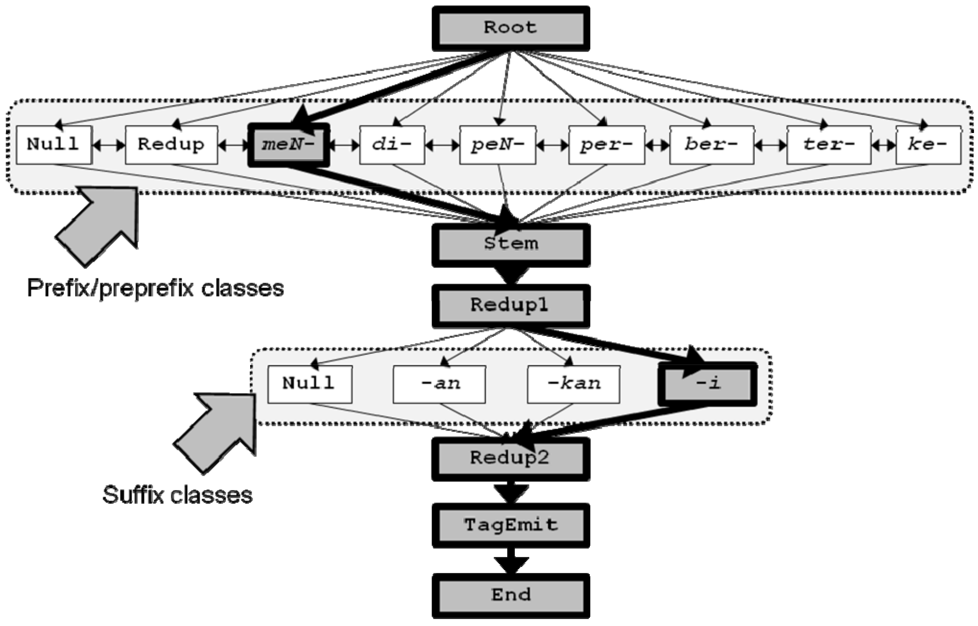


Figure 2. The path for *memukuli*

This rule represents a valid transition from the **SuffixI** state to the **Redup2** state by parsing the –i suffix and emitting the +**VERB** and +**AV** tags (**"+Verb":i "+AV":0**). Additionally, the diacritic tests **"@D.PREPREF@"** and **"@D.REDUP@"** stipulate that this transition can only be taken if the **PREPREF** and **REDUP** flags are not set, the disjunction **["@R.STEM.Vrb@"|"@R.STEM.Nom@"|"@R.STEM.Adj@"]** states that the **STEM** flag must be either **Vrb**, **Nom**, or **Adj**, and the **"@R.PREF.meN@"** test requires that the **PREF** flag is set to **meN**. Finally, **"@P.SUFF.i@"** states that the **SUFF** flag is set to **i**. Should any of these tests fail, the transition cannot be taken. If no other alternative paths are available to reach the **End** state, the word is rejected.

5.2 Reduplication Process

As discussed above in Sections 2 and 4, Indonesian morphology includes the non-concatenative process of reduplication. Handling this with pure regular grammars as implemented by finite state automata is very difficult. Thus, we employ the **compile-replace** feature in **xfst** (Beesley & Karttunen, 2000). This feature allows the repetition of arbitrarily complex sublanguages by specifying the brackets **"^["** and **"^]"** to mark the domain of reduplication. The right bracket is also augmented with **^2** to indicate duplication; thus, the full annotation is **"^["** and **"^2^]"**. Given this network definition, xfst compiles and post-processes these annotations to produce a new network where the appropriate reduplications have been carried out. For example, **"^[buku^2^]"** would eventually be compiled to become **bukubuku**.

Thus, the idea is to insert the **"^["** and **"^2^]"** annotations in the right places. Due to the various types of reduplication in Indonesian (see Section 2), the rules for reduplication can be found at the **Redup** (pre)prefix state and the **Redup1** and **Redup2** states. The **Redup** prefix state emits the opening **"^["** brackets and sets an appropriate flag as a reminder that the closing brackets are required. The **Redup1** state is responsible for closing partial and affixed reduplications, i.e. where the suffix is not included in the reduplication, whilst the **Redup2** state is responsible for closing full reduplication, i.e. where the suffix is part of the reduplication process. Both **Redup1** and **Redup2** states check for the value of the **REDUP** flag as set by the **Redup** prefix state.

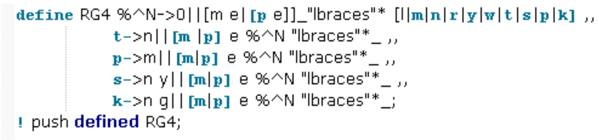
5.3 Morphophonemic rules implementation

The full transducer composes the morphotactic and morphophonemic rules. As a result, the output of the morphotactic rules implementation serves as the input to the morphophonemic rules implementation.

From the example in Section 5.1 regarding the implementation of morphotactic rules, it can be observed that the string *me^Npukuli* has been produced. However, the morphological analysis process is still incomplete since *me^Npukuli* is not a valid word in Indonesian. In order to complete the process, the morphophonemic rules must be implemented. Since the full transducer composes the morphotactic and morphophonemic rules, the word *me^Npukuli* becomes the input for the morphophonemic process.

The implementation of morphophonemic rules differs slightly from the implementation of morphotactic rules. For morphotactic rules, there are several steps that can be illustrated as a flow of process. However, the implementation of morphophonemic rules generally implies the rules itself. Each rule is defined as a replacement rule that will collaborate with other rules through composition or parallelization.

All rules in Section 4.4 have been implemented. However, due to space constraints, we will only explain the implementation of rule '**RG4**', which encodes */N/* deletion parallelized with four phonetic stem changes:



```
define RG4 %^N->0||[m e|[p e]]_"lbraces"* [l|m|n|r|y|w|t|s|p|k] ,,
          t->n||[m |p] e %^N "lbraces"*_ ,,
          p->m||[m|p] e %^N "lbraces"*_ ,,
          s->n y||[m|p] e %^N "lbraces"*_ ,,
          k->n g||[m|p] e %^N "lbraces"*_;
! push defined RG4;
```

This rule actually consists of five rules that are parallelized. The first rule is */N/* deletion, where the nasal symbol */^N/* is omitted if it is preceded by */me/* or */pe/* and followed by either of the phonemes */l/*, */m/*, */n/*, */r/*, */y/*, */w/*, */t/*, */s/*, */p/*, or */k/*.

The next four rules are the phonetic stem changes, which apply to stem words preceded by the morpheme */me^N/* or */pe^N/*, and whose initial phoneme is one of */t/*, */p/*, */r/*, */s/*, or */k/*. More specifically, the phoneme */t/* will transform into */n/*, */p/* will transform into */m/*, /s/ will transform into */ny/*, and finally the phoneme */k/* will transform into */ng/*.

All morphophonemic rules that implement the various processes are composed into one large rule, from which the morphophonemic transducer network can be constructed.

The morphophonemic rule that applies to the word *me^Npukuli* used as an example in Section 5.1 would be the above "*/N/* deletion parallelized with four phonetic stem changes", because the word *me^Npukuli* fulfils the requirements of that rule. As can be seen from the implementation, to get the "*/N/* deletion process" and "*/p/* replacement with */m/* process" ("*/p/* replacement with */m/* process" is one of those four phonetic rules), in that word there must appear the phoneme */^N/* preceded by */me/* or */pe/* and followed by phoneme */l/*, */m/*, */n/*, */r/*, */y/*, */w/*, */t/*, */s/*, */p/*, or */k/* and it also must have the phoneme */t/, /p/, /r/, /s/,* or */k/* preceded by the morpheme */me^N/* or */pe^N/*. Since the word *me^Npukuli* fulfils these requirements, */^N/* will be omitted and */p/* will transform into */m/* so that the word *me^Npukuli* will transform into the word *memukuli*. With the word *memukuli* being generated, the process is finished and the word *memukuli* is recognised as a valid word in Indonesian.

6 Evaluation

To assess our implemented system, we ran some test cases in the form of words extracted from an electronic version of the *Kamus Besar Bahasa Indonesia*[3]. We tested the implementations of the morphotactic and morphophonemic rules separately. To evaluate the ability of the analyser to both accept valid forms and reject invalid forms, the test cases included both valid and invalid morpheme combinations. Executing all of the test cases, we obtained the results shown in Table 1, which presents the results of morphotactic test cases, and Table 2, which presents the results of morphophonemic test cases. The 'Analysis' column displays the results of test cases where the surface form of an Indonesian word was provided as input, and our system was tasked with parsing the morphological structure. For example, given the word *memukul*, our system should return *pukul*+VERB+AV. On the other hand, the 'Synthesis' column concerns the opposite situation, i.e. test cases where the input is the string of morphological tags, and our system was tasked with generating the fully inflected form.

Result	Analysis	Synthesis	Total
1. Correct result	103	43	146
2. Several results, including correct	46	106	152
3. Incorrect result	3	3	6
Total	**152**	**152**	**308**

Table 1. Results of morphotactic test cases

Result	Analysis	Synthesis	Total
1. Correct result	51	21	72
2. Several results, including correct	6	36	42
3. Incorrect result	1	1	2
Total	**58**	**58**	**116**

Table 2. Results of morphophonemic test cases

We classify the test case results into three categories. The first category indicates that our system returned exactly one correct analysis or synthesis for a valid test case, or does not produce anything for an invalid test case. The second category is that, when given a valid test case, the system produces several answers, one of which is the desired result. Finally, the last category is seen when the system fails to analyse or synthesize a valid test case, or incorrectly produces an answer for an invalid test case.

From the tables, we can observe that the results for analysis are more accurate than for synthesis, where the system tends to produce more than one result. For example, our system correctly provides the single analysis for *memukul*. However, when trying to synthesize *pukul*+VERB+AV, it produces other alternatives, e.g. *memukulkan*, *memperpukuli*, *mengepukulkan*, etc. This suggests the need for a more refined set of morphological feature tags.

7 Summary

We have presented the design of an Indonesian morphological analyser that provides a detailed analysis of its rich affixation process using the two-level morphology approach, implemented using **xfst** and **lexc**. Our approach is able to handle reduplication, a non-concatenative morphological process. Our (not very rigorous) evaluation shows that the implementation is generally able to encode the rules of the various morphological processes.

[3] http://www.pusatbahasa.diknas.go.id/kbbi

References

Adriani, Mirna, Jelita Asian, Bobby Nazief, Seyed Mohammad Tahaghoghi and Hugh Williams. 2007. Stemming Indonesian: A Confix-Stripping Approach. *ACM Transactions on Asian Language Information Processing*, Vol. 6, No. 4.

Alwi, Hasan, Soenjono Sardjowidjojo, Hans Lapoliwa, and Anton Moeliono. 2003. *Tata Bahasa Baku Bahasa Indonesia Edisi Ketiga*. Pusat Bahasa dan Balai Pustaka, Jakarta.

Beesley, Kenneth and Lauri Karttunen. 2000. Finite-State Non-Concatenative Morphotactics. In *Proceedings of SIGPHON-2000*, August 2000, Luxembourg, pp 1-12.

Beesley, Kenneth and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publication, Stanford.

Dalrymple, Mary, Maria Liakata, and Lisa Mackie. 2006. Tokenization and Morphological Analysis for Malagasy. In *Computational Linguistics and Chinese Language Processing Vol.11, No.4, December 2006, pp 315-332*.

Gordon, Raymond (ed). 2005. *Ethnologue: Languages of the World 15th edition*. SIL International, Dallas, USA.

Hartono, Hendra. 2002. *Pengembangan Pengurai Morfologi untuk Bahasa Indonesia dengan Model Morfologi Dua Tingkat Berbasiskan PC-KIMMO*. Undergraduate thesis, Faculty of Computer Science, University of Indonesia

Jurafsky, Daniel and James Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Lingusitic, and Speech Recognition*. Prentice Hall, New Jersey.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Siregar, Neil. 1995. *Pencarian Kata Berimbuhan pada Kamus Besar Bahasa Indonesia dengan menggunakan Algoritma Stemming*. Undergraduate thesis, Faculty of Computer Science, University of Indonesia.

Sneddon, James. 2003. *The Indonesian Language: Its History and Role in Modern Society*. UNSW Press, Sydney, Australia.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 151-159

Punctuation normalisation for cleaner treebanks and parsers

Daniel Tse and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dtse6695,james}@it.usyd.edu.au

Abstract

Although punctuation is pervasive in written text, their treatment in parsers and corpora is often second-class.

We examine the treatment of commas in CCGbank, a wide-coverage corpus for Combinatory Categorial Grammar (CCG), re-analysing its comma structures in order to eliminate a class of redundant rules, obtaining a more consistent treebank.

We then eliminate these rules from C&C, a wide-coverage statistical CCG parser, obtaining a 37% increase in parsing speed on the standard CCGbank test set and a considerable reduction in memory consumed, without affecting parser accuracy.

1 Introduction

Although parsers must all at least accept text containing punctuation, there is much variation in how punctuation is treated during training, parsing and evaluation. The work of Nunberg (1990) brought attention to the fact that parsers can often reject otherwise ambiguous parses on the basis of punctuation, by considering a formal grammar of text units: chunks of text delimited by punctuation tokens.

In this work, we explore how the treatment of commas, the most common class of punctuation in the Penn Treebank (Marcus et al., 1994), affects their analysis in CCGbank (Hockenmaier and Steedman, 2007), a 1.2 million word corpus for Combinatory Categorial Grammar (CCG), generated from a subset of the Penn Treebank. We uncover a systematic inconsistency in the way CCGbank represents sentences involving commas, and perform a transformation of the original corpus which eliminates this source of uninformative variation, resulting in a version of CCGbank which assigns a uniform structure to each of the syntactic roles played by commas in the corpus.

By removing the superfluous rules induced by this inconsistency from the corpus, we obtain cleaner analyses of comma structures which require fewer CCG rules to explain. These changes to the corpus, in turn, allow us to simplify the implementation of the C&C parser (Clark and Curran, 2007), a wide-coverage statistical CCG parser for English, by barring the parser from considering the CCG rules which we have made redundant.

Training the C&C parser on our modified CCGbank yields average speed gains of 37% and 21% on the standard CCGbank test set and development set respectively, and a 47% reduction in memory consumed by the chart parsing process as evaluated on the standard CCGbank test set. Parser coverage increases slightly while accuracy is not affected.

Consistency, a powerful and general guideline in corpus design, can improve the usefulness and quality of corpora and the applications built with them. We eliminate a source of systematic inconsistency in comma representation from a wide-coverage corpus and parser, resulting in cleaner versions of both resources. The combination of a cleaner treebank and parser leads to gains in speed and a reduction in memory consumption without affecting parser accuracy.

2 Background

Nunberg (1990) characterises the traditional view of written language as a simulacrum of a richer medium, that of spoken language. Nunberg notes that although formal prescriptive accounts of punctuation have long been the domain of grammarians (and the bane of their students), this perception of writing as an inferior or merely transcriptional medium has lead to the relative paucity of analytic accounts of punctuation in linguistics.

Nunberg (1990) describes the macro-structure of English text as two superposed systems: the *text grammar* defines the well-formedness of a sentence interpreted as a sequence of not necessarily constituent-forming units, while the *lexical grammar* encodes the usual relationships between individual lexical items.

Nunberg's validation of punctuation as a linguistic system meriting study in its own right spurred work affording punctuation a more integral role in generation and parsing (Briscoe, 1994; Osborne, 1995), as well as work in corpus linguistics as to the distribution and semantics of commas, and other classes of punctuation tokens (Bayraktar et al., 1998; Jones, 1997).

However, despite this renewal of interest, punctuation is still often relegated to a marginal role in parsing and the evaluation of parsers, because its representation is often inconsistent between treebanks, and even within a treebank. `evalb`, a widely used script for calculating the crossing-brackets parser evaluation metric PARSEVAL, strips all punctuation from the candidate and gold standard text (Sekine and Collins, 2006), so that the attachment level of any punctuation does not influence the bracketing metric.

Briscoe (1994) discusses a parsing system which uses Nunberg's concept of a text grammar to inform its chunking decisions (in this context, the segmentation of input text into sentences, robustly handling non-terminal uses of periods, for example). In Briscoe's work, an implementation of text grammar can eliminate incorrect parses based on punctuation. In Example 1, a variation on the well-known *PP* attachment problem, a parser which simply ignores punctuation in the input will encounter ambiguity which a punctuation-aware parser will not.

(1) I saw the girl on the hill with the telescope, from the store.

(2) I saw the girl on the hill with the telescope from the store.

The work of Djordjevic et al. (2007) for the C&C parser constrains phrases delimited by hyphens, colons and semicolons to be complete constituents, allowing the parser to avoid great amounts of attachment ambiguity when parsing long sentences. However, while Djordjevic et al. modify the parser to derive information from punctuation in the corpus, this work changes the representation of commas in the corpus to convey information on each comma's role to the parser. The corpus we obtain assigns a uniform structure to each of the comma's syntactic roles.

3 CCG and CCGbank

In Combinatory Categorial Grammar (Steedman, 2000), each lexical item receives a *category* such as N/N or $(S\backslash NP)/NP$, which determines how it may combine with other groups of lexical items. New categories may be formed of adjacent categories by applying *combinatory rules*. In Figure 1, each line denotes the application of a particular combinatory rule, which combines at most two categories into another category. Combinatory rules enable succinct, natural CCG analyses for difficult constructions such as argument cluster coordination and parasitic gaps (Steedman, 2000).

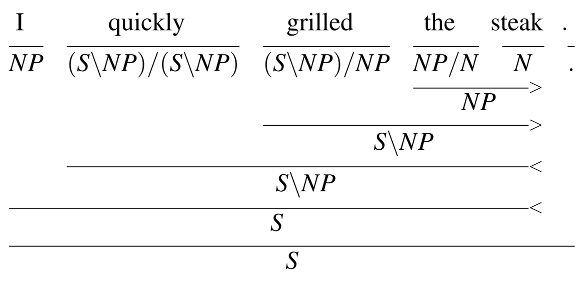


Figure 1: CCG analysis

CCGbank is a wide-coverage CCG corpus, generated from the Wall Street Journal section of Penn Treebank through an automatic conversion procedure described by Hockenmaier and Steedman (2007). CCGbank enabled the development of wide-coverage statistical CCG parsers such as the C&C parser Clark and Curran (2007), on which we will evaluate this work. Parsing in the C&C parser proceeds in two broad stages: a maximum entropy *supertagger* assigns categories to lexical items, which the *parser* then attempts to combine using the CCG rules. The corpus, as a collection of CCG derivations, implicitly encodes the set of CCG rules used in its derivations. These rules must be explicitly encoded in a parser trained on CCGbank, since the training process requires it to reproduce the derivations represented in the corpus. The goal of this work is to drastically reduce the number of rules implicitly encoded by CCGbank, and in turn reduce the number of rules explicitly encoded in the C&C parser.

The distribution of punctuation symbols in CCGbank is given in Figure 2. Commas account for over half of all punctuation tokens, and periods well-represented but not as common as commas. (Bayraktar et al., 1998) observes the same dramatic drop between the frequency of commas and periods, and the remaining punctuation marks.

Symbol	Frequency	
comma	59991	(53.77%)
period	47875	(42.91%)
colon	1649	(1.48%)
semi-colon	1460	(1.31%)
question mark	511	(0.46%)
excl. mark	71	(0.06%)
apostrophe	3	(0.002%)
Total punct	111560	

Figure 2: Punctuation distribution in CCGbank

Although periods approach commas in frequency, their representation in CCGbank is largely consistent, since all of their roles are constituent-final: sentence terminator, list index delimiter, abbreviation marker. By comparison, commas in CCGbank occur both constituent-initially as well as finally.

The goal of this work is to standardise the representation of commas in CCGbank, either by converting all *left commas* (comma leaves which are the left child of their parent) to *right commas*, or vice versa, and in doing so, allow us to remove unnecessary rules from the parser. In the next section, we discuss and justify which of the CCGbank rules are unnecessary, and which CCGbank rules our normalisation procedure should not modify.

4 Normalising CCGbank

Our goal of eliminating uninformative variation from the corpus requires us to distinguish cases where comma direction conveys some useful information from those where it has no discriminative function. We partition the CCGbank rules involving a comma argument so that our processing only affects those cases we are free to manipulate.

The intuition for our transformations on CCGbank are as follows: commas are by a large margin, the most common form of punctuation encountered in Penn Treebank (Bayraktar et al., 1998).

The inconsistency in the representation of commas is an artifact of the procedure by which Penn Treebank derivations are transformed into CCGbank derivations. Given a Penn Treebank derivation P, the procedure uses head-finding heuristics to identify the head of each constituent. Binarisation is a consequence of the fact that CCG's combinatory rules are all unary or binary, requiring us to transform the k-way branching structures of Penn Treebank-style annotation. The procedure uses the heads identified by the first step to determine whether to generate left- or right-branching structures: constituents left of the head are left-branching, and those right of the head branch rightwards, as depicted in Figure 3. This ensures that adjuncts seek the head, and not vice versa.

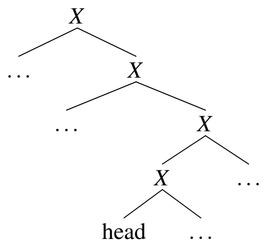


Figure 3: CCGbank binarisation

Accordingly, whether or not a comma node is a left, or a right comma, depends on whether it was left or right of the node identified by the head-finding heuristic as the head, as shown in Figure 4.

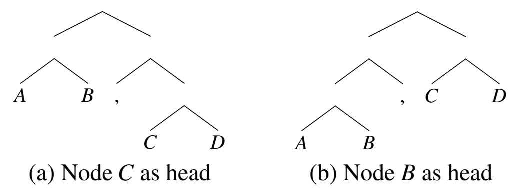


Figure 4: Left and right commas

This means that for many categories, the corpus contains a pair of *absorption rules*, so called because they leave the input category X unchanged:

$$X \quad , \quad \rightarrow \quad X$$
$$, \quad X \quad \rightarrow \quad X$$

which, by virtue of being extra-combinatory rules, must each be encoded in the parser. This proliferation of extra rules is merely an artifact of the binarisation process; in these cases, we judge that there is no evidence for the comma belonging to one side or the other, and that we are free to process the corpus so that it embodies only one of the above two rules.

4.1 Type-change rules

Type-change rules in CCGbank analyse syntax such as apposition without creating additional categorial ambiguity. The tradeoff is that such rules are completely unlicensed by the underlying theory, hence the designation *extra-combinatory*. Hockenmaier and Steedman (2007) argue for the use of extra-combinatory rules, to mitigate against the sparsity of data which would be entailed by giving every apposition *NP* a different category to an *NP* not participating in apposition. For example, the CCGbank analysis of sentence-final *NP* apposition:

(3) The index fell to 997, the first decline since July.

involves the following comma type-change rule:

$$, \quad NP \quad \rightarrow \quad (S\backslash NP)\backslash(S\backslash NP) \quad (\mathbf{T},)$$

The rule specifies that an *NP* following a comma can be treated as a verb phrase modifier (in general, the category of a modifier is X/X or $X\backslash X$ for some category *X*), allowing the analysis shown in Figure 5.

In CCGbank, the comma introducing an apposition is necessary to the resulting analysis, in that the rule does not trigger unless the comma is present. As such, comma type-change rules are fundamentally different from the comma absorption rules seen above, which neither introduce dependencies, nor affect interpretation. Furthermore, we can see that the above rule cannot involve anything but a left comma, for the simple reason that it is the analysis for a sentence-*final NP* apposition. Therefore, if we were to blindly normalise the above rule to:

$$NP \quad , \quad \rightarrow \quad (S\backslash NP)\backslash(S\backslash NP) \quad (\mathbf{T},)$$

we would no longer attain the same analysis of sentence-final *NP* apposition. In general, Bayraktar et al. (1998) consider the commas which introduce apposition to be a form of paired punctuation, like quotes or parentheses. Unlike close quotes, or close parentheses, however, the ‘closing comma’ is suppressed when it immediately precedes the sentence-final period. This suggests that it should be the ‘opening comma’ that triggers the use of type-change rules enabling apposition, since the ‘closing comma’ is not realised sentence-finally, as shown in Example 5.

(4) Pierre Vinken, chairman of Elsevier, will become non-executive director.

(5) The non-executive director is Pierre Vinken, chairman of Elsevier.

Hence, our comma normalisation does not process commas involved in comma type-change rules.

4.2 Conjunction commas

CCG is known for its straightforward analysis of coordination: two categories may be coordinated with a conjunction precisely when the conjuncts have the same category (Steedman, 2000). The CCGbank treatment of coordination differs from the rule of syncategorematic coordination originally given by Steedman (2000), because of the undesirability of introducing such a ternary rule as an exception to a system of otherwise unary and binary combinatory rules. To analyse coordination of nouns in series:

(6) Pound cake consists of butter, eggs and flour.

CCGbank instead provides the following pair of extra-combinatory rules:

$$, \quad N \rightarrow N[conj] \qquad (\Phi_1)$$
$$N \quad N[conj] \rightarrow N \qquad (\Phi_2)$$

which yield an analysis isomorphic to Steedman's ternary coordination rule, without the need to directly implement such a ternary rule.

A category of the form $X[conj]$ represents a conjunct which has picked up a conjunction word, but not the other conjunct[1].

[1] In the analysis of Figure 6, a comma inserted before ‘*and*’ (the so-called *Oxford comma*) would receive the comma category , and be subject to comma absorption, with the lexical conjunction ‘*and*’ serving as the actual conjunction word.

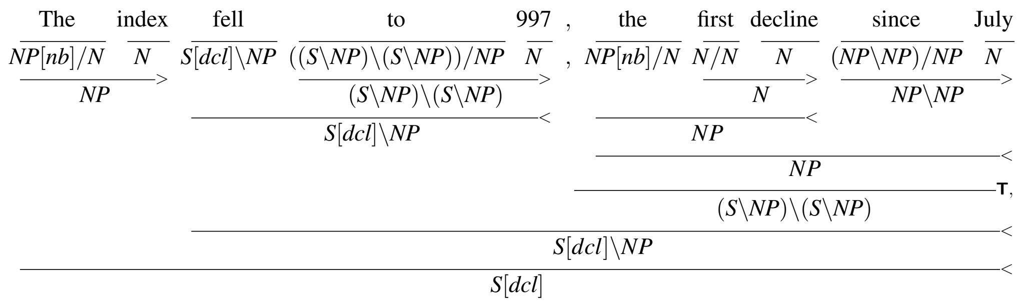


Figure 5: The type change rule $, \ NP \rightarrow (S\backslash NP)\backslash(S\backslash NP)$ is applied in the fifth row

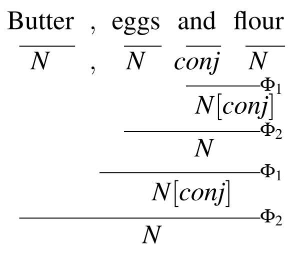


Figure 6: CCGbank analysis of coordination

For example, the partial coordination *and flour* in Figure 6 receives the category $NP[conj]$.

Suppose we are to convert the left comma rule Φ_1 to a right comma rule as a candidate for normalisation.

$$N \quad , \quad \rightarrow N[conj] \qquad (\Phi'_1)$$

Such a rule has the effect of consuming conjuncts left-to-right, instead of right-to-left in the canonical set of rules. To support such an order, we would also need to change rule Φ_2 accordingly:

$$N[conj] \quad N \rightarrow N \qquad (\Phi'_2)$$

That is, while the canonical set of rules yields a right-branching analysis of coordination, replacing the left comma rule with a right comma rule yields a left-branching analysis instead. Functionally, there is no difference between a left- and right-branching analysis. However, if we were to convert the left comma coordination rule Φ_1 to a right comma rule, we would have to re-analyse all right-branching coordination structures in the corpus as left-branching structures, with no change in semantics. Furthermore, we would save the parser no effort, since the original rule Φ_1 does not have a right comma rule analogue which we could eliminate. Therefore, we exclude coordination comma rules as candidates for comma normalisation.

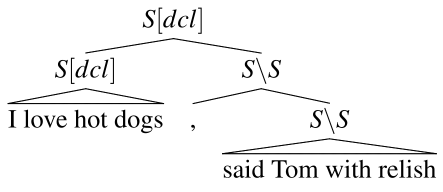
4.3 Left, or right commas?

We now have a choice between normalising all absorption commas to left commas or to right commas. We have seen that the two non-absorption classes of comma rules both involve left commas, while absorption comma rules occur both as left and right comma rules. We make the arbitrary choice to normalise all absorption comma rules to right comma rules, so that it is easier for corpus applications to distinguish the most common case of punctuation absorption from other "special" rules such as type-changing and coordination.

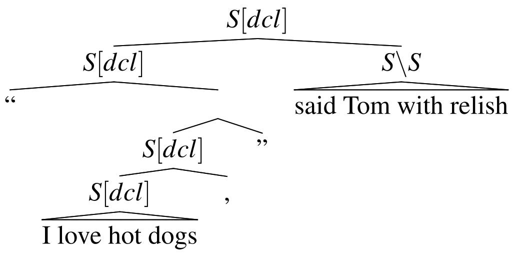
Although an additional reason for normalising to right commas rather than left commas is not evident from our analysis of the *unmodified* CCGbank, an argument arises when we consider the version of CCGbank described by Tse and Curran (2007), which restores quote symbols to CCGbank. The requoting procedure operates by recovering the positions of open and close quotes from CCGbank's source corpus, Penn Treebank. If the text lying between the start and end of the quoted span exactly spans a subtree, then we can enclose that entire subtree in a quoting structure. Otherwise, we attach the open and close quotes separately. However, a common convention in English prose style is to include the comma which offsets a span of quoted direct speech as part of the span of quoted text.

(7) "I love hot dogs," said Tom with relish.

In the unmodified corpus, such a comma is attached to the right subtree, as in Figure 7a. However, this prevents us from enclosing the span of quoted text (including the comma), since the comma is stranded in the right-hand subtree. Accordingly, we must transform such a left comma to a right comma in order to obtain an analysis in which the open and close quotes correctly enclose the span of quoted text, as in Figure 7b.



(a) Original analysis



(b) Modified analysis

Figure 7: Quote insertion and comma direction

The fact that the correct analysis of such quoting structures *requires* right comma absorption suggests that for consistency, we should normalise to right, rather than left commas.

To summarise, our criteria for normalisation are to process only absorption commas, leaving type-change comma rule instances and analyses of coordination unmodified. The normalisation of all absorption commas to right commas, which enables us to remove all left comma absorption rules from the parser, allows us to make accuracy gains and speed reductions in the parser.

We also hypothesise that no useful information is lost as long as we restrict comma normalisation to the absorption cases we have identified above, since commas involved in absorption rules do not project dependencies, and hence cannot adversely affect the standard dependency-based CCGbank evaluation we describe in Section 7.

5 Transformations

Having shown that we are free to manipulate instances of absorption comma rules, the transformations themselves are simple. The procedure for comma normalisation is given in Algorithm 1.

We choose to re-attach the comma node as high as possible in the derivation tree, as shown in Figure 8b, in the absence of a more sophisticated

Algorithm 1 NORMALISE-COMMAS(C):

```
for each leaf $l$ in $C$ do
  if $(l, l.sibling \to l.parent)$ is comma absorption then
    $cur \leftarrow l$.parent
    Delete $l$ and its sibling
    {Re-attach the comma as high as possible}
    while $cur$ is the left child of its parent do
      $cur \leftarrow cur$.parent
    end while
    Insert comma structure at sibling of $cur$
  end if
end for
```

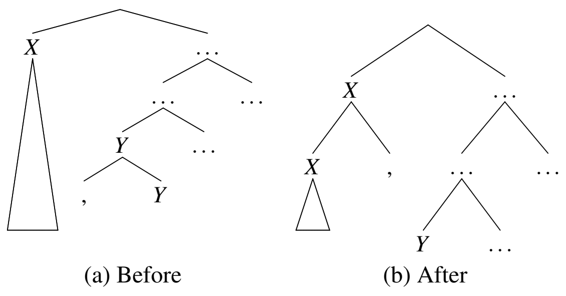


Figure 8: Comma movement

method of determining the depth in the left subtree at which the comma should be restored. With reference to Figure 8, the algorithm has converted every instance of a left absorption rule $(, \quad Y \quad \to Y)$ to an instance of some right absorption rule $(X \quad , \quad \to X)$. Performing this transformation on every absorption comma in the corpus achieves the desired normalisation.

Figure 9 shows the distribution of occurrences of each kind of comma rule before, and after comma normalisation. The frequencies for two CCG rules for coordination which differ in whether a comma, or a lexical conjunction is used as the coordinating word are aggregated in the second row.

Our procedure successfully re-analyses every left absorption comma in CCGbank as an instance of a right absorption rule. The two instances of left commas which remain in the comma normalised corpus result from mis-tokenisations in Penn Treebank: a numeral and a determiner mis-tagged as a comma.

As a final step, we remove support for the now-eliminated left comma rules from the C&C parser,

CCG rule	Before	After
$, \quad X \quad \rightarrow X$	25690	2
$X \quad , \quad \rightarrow X$	21990	47678
$\{,\|conj\} \quad X \quad \rightarrow X[conj]$	11466	11466
Other typechange	607	607
$, \quad NP \rightarrow (S\backslash NP)\backslash(S\backslash NP)$	242	242
Total	*59995*	*59995*

Figure 9: Number of comma rule instances before and after normalisation

preventing the parser from hypothesising the attachment of commas to the left in the absorption case, reserving left-attachment analyses for the special cases of coordination and type-change structures.

6 Evaluation

We wish to determine the impact of our changes in two respects: their effect on the accuracy of the resulting parser model and on parser ambiguity, relative to the unmodified parser and corpus.

To measure the impact of the modifications on parser accuracy, we performed the dependency-based evaluation of Clark and Curran (2004) on the standard development and test sets, sections 00 and 23 of CCGbank. We evaluate the performance of a parser and model by computing the F-score over the obtained set of predicate-argument dependencies, relative to a gold standard. In the *labelled* CCG evaluation, a dependency consists of a tuple $\langle H, C, i, D, l \rangle$, where H is the head lexical item, C is its category with its arguments annotated with indices, i is the index of the argument satisfied by this dependency, and D is the lexical item satisfying this dependency. l is a flag distinguishing local from long-range dependencies. The *unlabelled* evaluation is performed against reduced tuples $\langle H, D \rangle$. Additional evaluation metrics are *sentence level accuracy*: the proportion of sentences for which the parser produced every gold standard dependency, *category accuracy*: the proportion of correctly assigned categories, and *coverage*: the proportion of sentences for which the parser obtained some spanning analysis. The column *Auto F* denotes the labelled F score when the C&C parser assigns its own POS tags instead of using gold standard tags.

To gauge the impact on parser ambiguity and memory consumption, we consider the number of *conjunctive nodes* generated by the C&C parser during the parsing process. The C&C parser uses a *packed chart representation* to reduce the size in memory of the parsing chart by allowing partial derivations which span the same category and have the same set of unfilled dependencies to occupy a single cell in the chart. A conjunctive node is formed when two single cells in the chart are merged into another cell, as occurs when two categories are being combined. Accordingly, the number of conjunctive nodes is an indication of the quantity of category combinations performed by the parser, and hence the degree of ambiguity encountered in parsing a given sentence as well as the physical size in memory of the chart data structure (Clark and Curran, 2007).

We also perform the standard evaluation 25 times on the development and test sections to compute the average wall-clock time for each experiment.

For the sake of comparison, we also perform the above experiments using our transformed corpus, but without barring the consideration of the now-redundant rules in the parser. In the results table, we refer to this ensemble as *New**.

7 Results

Figure 10 shows that the comma-normalised corpus, in concert with our modifications to the CCG parser removing support for the now-redundant rules, outperforms the baseline in ambiguity and parsing speed without adversely affecting parser performance or corpus coverage. This supports our claim in Section 4 that modifying only absorption commas does not remove any useful information from the corpus.

Why does comma normalisation have such a strong effect on parser efficiency? Any absorption rule engenders considerable ambiguity: consider that a comma can be attached at any internal level in the derivation. Although the C&C parser reins in this ambiguity by restricting the categories with which punctuation can combine, the effect of this attachment ambiguity is prominent when we consider the number of CCGbank sentences containing punctuation tokens apart from the full stop (32636 of 48934, or 66.7% of CCGbank derivations). Our changes to the parser reduce the number of comma absorption rules from 19 to 9, considerably reducing parser ambiguity in these sentences.

		Labelled			Auto	Sent.	Unlabelled			Cat.	Covg
	Experiment	*P*	*R*	*F*	*F*	acc.	*P*	*R*	*F*	acc. %	%
Dev. set:	*New*	85.60	84.79	**85.19**	**83.39**	31.65	92.40	91.52	**91.96**	**93.08**	**99.11**
	Baseline	85.52	84.69	85.11	83.36	**31.93**	92.37	91.47	91.92	93.02	99.06
Test set:	*New*	86.17	85.48	**85.82**	**83.45**	**35.14**	92.33	91.59	**91.96**	**93.38**	**99.67**
	Baseline	86.03	85.37	85.70	83.24	34.70	92.28	91.57	91.93	93.27	99.63

(a) Standard CCGbank evaluation on development and test sets

		Average	Parsing rates		Avg.	Avg.
	Experiment	parsing time	*sents/s*	*words/s*	conj. nodes	total nodes
Dev. set:	*New*	**89.49s**	**21.38**	**507.61**	**2472.77**	**5260.84**
	*New**	109.97s	17.40	413.05	3446.41	7800.58
	Baseline	112.73s	16.97	402.93	3349.94	7662.95
Test set:	*New*	**56.28s**	**42.78**	**984.05**	**2112.78**	**3653.83**
	*New**	88.30s	27.26	627.08	3087.54	6779.42
	Baseline	89.32s	26.95	619.97	3010.34	6590.26

(b) Parsing time on development and test sets

Figure 10: The effects of comma normalisation on parser accuracy and speed

An unexpected result is that the new corpus trained on the old parser (experiment *New**) results in *greater* ambiguity compared to the baseline. We determined that this is caused by our naïve choice of comma attachment level (as high as possible in the tree, the most general position). Our transformations have inadvertently added a small number of new rules to the corpus (with reference to Figure 8, this will occur when $(X\ ,\ \rightarrow X)$ is not attested in the corpus). On examination, in all of these newly added rules, the non-comma argument is the result of a *unary type-change* rule (such as $NP \rightarrow S/(S\backslash NP)$). We note by way of future work that we can eliminate these added rules in this way: if attaching as high as possible (our current criterion) would yield such a new rule, then attach the comma at its child instead (before such a rule has been applied).

We have developed a comma-normalised corpus which explains the same data with fewer rules, while slightly improving the coverage, accuracy and parsing time of a parser trained on this improved corpus. The resulting version of CCGbank treats the ubiquitous comma consistently, a desirable attribute for any NLP task which uses this valuable resource.

8 Conclusion

We believe that further improvements in parsing speed and memory consumption are possible by changing the representation of comma structures in the corpus. As we observed in Section 7, when the now-redundant rules are not disabled, parser ambiguity actually slightly *exceeds* the baseline. Changing the comma attachment level criterion may further improve parser ambiguity and memory consumption by reducing the number of rules the transformation adds to the treebank.

Far from being second-class, the correct analysis and treatment of punctuation in corpora and parsers has practical ramifications. We would like to continue to explore punctuation-awareness in parsing in the vein of Djordjevic et al. (2007), but from the viewpoint of corpus design. Can we bring the benefits of Nunberg's text grammar to a regular treebank by superimposing text grammar constituents onto those of the usual lexical grammar?

We would also like to explore the cross-linguistic treatment of punctuation in treebanks to inform possible improvements to existing corpora and derive guidelines for the design of future corpora.

We have eliminated a source of systematic inconsistency in a wide-coverage CCG treebank and simplified the implementation of a CCG parser, obtaining considerable improvements in speed and memory usage without sacrificing parser accuracy, demonstrating the importance of principled, consistent annotation in corpus design.

References

Murat Bayraktar, Bilge Say, and Varol Akman. 1998. An Analysis of English Punctuation: The Special Case of Comma. *International Journal of Corpus Linguistics*, 3(1):33–57.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britt Schasberger. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. *University of Pennsylvania, Philadelphia*.

Ted Briscoe. 1994. Parsing (with) Punctuation etc. *Research Paper, Rank Xerox Research Centre, Grenoble*.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 104–111. Association for Computational Linguistics, Barcelona, Spain.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Bojan Djordjevic, James R. Curran, and Stephen Clark. 2007. Improving the Efficiency of a Wide-Coverage CCG Parser. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 39–47. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2005. CCGbank: Users manual. *University of Pennsylvania, Philadelphia*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Bernard Jones. 1997. *What's the Point? A (Computational) Theory of Punctuations*. Ph.D. thesis, PhD thesis, Centre for Cognitive Science, University of Edinburgh, Edinburgh, UK.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. Center for the Study of Language and Information.

Miles Osborne. 1995. Can punctuation help learning? *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, Lecture Notes in Artificial Intelligence*, pages 399–412.

Satoshi Sekine and Michael J. Collins. 2006. Evalb: a parseval crossing brackets evaluation script. URL `http://www.cs.nyu.edu/cs/projects/proteus/evalb`, accessed 10 June 2008.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press. Cambridge, MA, USA.

Daniel Tse and James R. Curran. 2007. Extending CCGbank with quotes and multi-modal CCG. *Australasian Language Technology Workshop 2007*, pages 149–151.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 160-168

Generating Relational References: What Makes a Difference?

Jette Viethen
Centre for Language Technology
Macquarie University
Sydney, Australia
jviethen@ics.mq.edu.au

Robert Dale
Centre for Language Technology
Macquarie University
Sydney, Australia
rdale@ics.mq.edu.au

Abstract

When we describe an object in order to enable a listener to identify it, we often do so by indicating the location of that object with respect to other objects in a scene. This requires the use of a *relational referring expression*; while these are very common, they are relatively unexplored in work on referring expression generation. In this paper, we describe an experiment in which we gathered data on how humans use relational referring expressions in simple scenes, with the aim of identifying the factors that make a difference to the ways in which humans construct referring expressions.

1 Introduction

The generation of referring expressions—the process of determining how best to describe an object in order to enable the hearer to identify it—is a widely explored topic in natural language generation research. We expect the first practical applications of the algorithms developed in this area to be those where the location of objects within scenes is a requirement: examples of such scenarios are the description of entities such as buildings and other landmarks in automatically-generated route descriptions (see Dale et al. (2005)) and the description of locations in 'omniscient room' scenarios, where an intelligent agent might try to tell you where you left your RFID-tagged keys. In these scenarios, it is very likely that the referring expressions generated will need to make use of the spatial relationships that hold between the intended referent and other entities in the domain; but, surprisingly, the generation of relational references is a relatively unexplored task. The few algorithms that address this task (Dale and Haddock (1991), Gardent (2002), Krahmer and Theune (2002), Varges (2005), Kelleher and Kruiff (2005, 2006)) typically adopt fairly simple approaches: they only consider spatial relations if it is not possible to fully distinguish the target referent from the surrounding objects in any other way, or they treat them in exactly the same as non-relational properties. As acknowledged by some of this work, this creates additional problems such as infinite regress and the inclusion of relations without regard for the properties of the landmarks that are associated with them.

To be able to develop algorithms that meet the requirements of applications like those just mentioned, we first need to have a better understanding of how humans use relational referring expressions. In this paper, we provide a detailed analysis of a set of experimental data that was gathered in a context where human subjects were encouraged to use relational expressions in order to identify intended referents.

In Section 2 we describe our data gathering experiment in some detail, explaining the rationale behind our development of the test data we used. In Section 3 we provide a summary of the corpus of referring expressions generated by the experiment. Section 4 provides a detailed analysis of the relational referring expressions in the corpus, identifying a range of phenomena of interest. Section 5 concludes the paper by drawing out some desiderata for the development of referring expression generation algorithms that follow from these observations.

2 The Data Gathering Experiment

2.1 General Overview

The real contexts in which referring expressions are used can be very complex. Consider the following hypothetical references in the motivating scenarios we used in the introduction:

(1) Turn left after the second shopfront that has a 'For lease' sign in the window.

(2) Your keys are under the loose leaf folder on the desk in the upstairs study.

In such real life situations, there are generally too many variables to permit carefully controlled experiments that would allow us to derive general principles for content determination. In line with almost all work in this area (see, for example, Brennan and Clark (1996), Thompson et al. (1993), Gorniak and Roy (2004), Jordan and Walker (2005), Byron and Fosler-Lussier (2006)), we therefore begin our explorations with very much simpler scenarios that allow us to explore specific hypotheses and to characterise the general strategies that humans seem to adopt; we can then apply these strategies in more complex scenes to see whether they continue to be applicable.

Our goal is to determine what characteristics of scenes impact on the use of spatial relations. The data gathering experiment we conducted had the form of a self-paced on-line language production study. Participants visited a website, where they first saw an introductory page with a set of simple instructions and a sample stimulus scene. Each participant was assigned one of two trial sets of ten scenes each. The scenes were presented successively in a preset order. Below each scene, the participant had to complete the sentence *Please pick up the* ... in a text box before clicking on a button to see the next scene. The task was to describe the target referent in the scene (marked by a grey arrow) in a way that would enable a friend looking at the same scene to pick it out from the other objects.

74 participants completed the experiment. They were recruited by emailing self-reported native English speakers directly and asking them to pass on the invitation for participation. The participants were from a variety of different backgrounds and ages, but were mostly university-educated and in their early or mid twenties. For reasons outlined in Section 3.1, the data of 11 participants was discarded. Of the remaining 63 participants, 29 were female, while 34 were male.

2.2 Stimulus Design

2.2.1 The Components of Scenes

In order to explore even the most basic hypotheses with respect to the use of relational expressions, we require scenes which contain at least three objects. One of these is the intended referent, which we refer to as the *target*. The subject has to describe the target in such a way as to distinguish it from the other two objects in the scene. Although our scenes are such that spatial relations are never *necessary* to distinguish the target, the scenes are set up so that one of the two non-target objects was clearly closer to the target. We call this object the (potential) *landmark*; the third object in the scene is then the *distractor*.

To minimise the number of variables in our experiments, we restrict ourselves to only two kinds of objects, cubes and balls. The objects also vary in two dimensions: colour (either green, blue, yellow, or red); and size (either large or small).

To reduce the number of factors in our scene design, the landmark and distractor are always placed clearly side by side, and the target is located on top of or directly in front of the landmark. This results in four possible spatial configurations:

1. target on top of landmark, distractor to the left;
2. target on top of landmark, distractor to the right;
3. target in front of landmark, distractor to the left; and
4. target in front of landmark, distractor to the right.

Whether the distractor is to the left or to the right of the other two objects determines what we call the *orientation* of the scene.

2.2.2 Creating a Balanced Stimulus Set

Together, the apparently simple factors described above allow 16,384 distinct scenes to be constucted. Clearly this is too many for us to experimentally determine generalisable observations, so we took additional measures to reduce this number while keeping the stimulus set as balanced as possible.

Firstly, we decided to only use two colours in each scene, with the result that, in any scene, at least two objects will have the same colour. The literature on visual salience (for an overview, see Chapters 1 and 2 of Pashler (1998), Yantis and Egeth (1999), and Caduff and Timpf (2007)) suggests that colour salience is relative rather than absolute. Consequently, we did not expect the individual colours to influence which objects were included in referring expressions; rather, we would expect that this is influenced by how different the colour of each object is from the colours of the other objects. We therefore attempted to keep the brightness of the different colours in each scene as constant as possible by choosing one of two colour 'templates' for each scene: blue+green and red+yellow. In order to make the experiment less monotonous for subjects, half of the scenes use one colour template, and the other half use the other.

Similarly, we would not expect the *orientation* of the scene to have an impact on the use of relations; so again we switch the orientation in half of the scenes in order to reduce monotony for the subject. However, having both orientations and both colour templates in the stimulus set still allows us to test for any unexpected impact of these factors.

Secondly, we chose to make all landmark objects cubes, simply because it might look unnatural to see an object balanced on top of a perfectly spherical ball. We also decided to exclude variation in the size of the target object from our analysis, making all target objects small; this avoids having to deal with the complexity of situations where the target object might obscure a smaller object which could otherwise be used as a landmark.

From the outset we had excluded situations in which spatial relations would be necessary to fully distinguish the target, i.e. scenes in which the target is identical to one or both of the other objects.

Even with these constraints, we still have 244 possible scenes. This is still too many to allow each possible scene to be tested a reasonable number of times without overloading our subjects; our aim, therefore, was to arrive at 20 stimulus scenes that could be divided into two equivalent trial sets, so that each participant would only have to describe 10 objects.

2.2.3 Scene Schemata

Our next step was to create five *schemata* (see Figure 1) as a basis for our final stimulus set. A schema

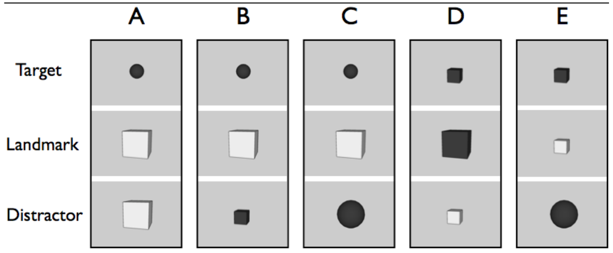


Figure 1: The *schemata* which form the basis for the stimulus scenes.

determines the type and size of each object in the scenes that are based on it, and determines which objects share colour. So, for example, in scenes based on Schema C, the target is a small ball; the landmark is a large cube; the landmark has a different colour from the target; and the distractor is a large ball sharing its colour with the target.

The design of these schemata is informed by three initial questions that we would like to explore:

1. Is the decision to use a spatial relation impacted by the length of the *minimal non-relational description* for the target?

A minimal non-relational description is the shortest referring expression that uniquely describes the target referent without using any relations to other objects or to the scene as a whole. To test this we need to have scenes where the target can be distinguished only by its type (Schemata A and B), scenes where a combination of type with either colour or size suffices to describe the target (Schemata C and E), and scenes where all three non-relational properties are necessary (Schema D). This question is examined in Section 4.3.1.

2. Is the use of spatial relations impacted by the similarity between target and landmark?

In Schemata A, B and C, these two objects share no properties; and in Schemata D and E, they share two properties, which is as similar as they can be without sharing all their properties. We look at the influence of target–landmark similarity in Section 4.3.2.

3. Is the use of spatial relations impacted by the similarity between landmark and distractor?

Based on the assumption that the salience of the landmark might have an impact, and that its salience is in turn determined by its similarity to the other

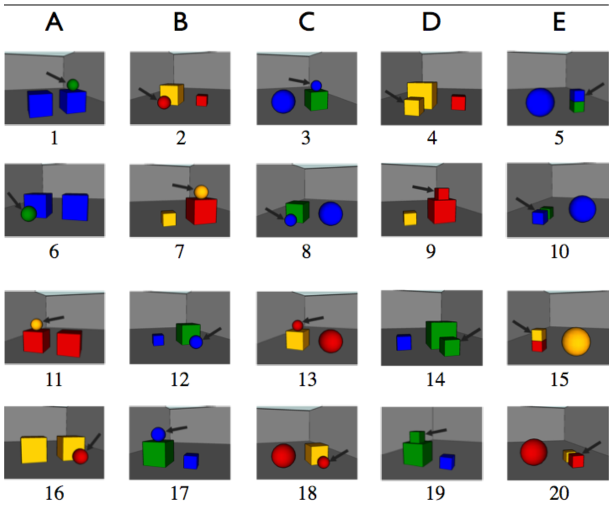


Figure 2: The stimulus scenes. The letters indicate which schema from Figure 1 each column of scenes is based on.

two objects, we were mindful that the stimuli should be also somewhat balanced regarding the landmark's similarity to the distractor. In Schema A, landmark and distractor are identical; in Schema E, they are completely distinct from each other; and in Schemata B, C and D, they share only one property value (their size in Schema C, and their type in Schemata B and D). The effect of landmark–distractor similarity is examined in Section 4.3.3.

2.2.4 Deriving Scenes from Schemata

From each schema we generated four scenes, resulting in the 20 stimulus scenes shown in Figure 2.

First we created Scenes 1–5, each based on a different one of the five schemata, by alternating between the colour templates and the spatial relations between target and landmark and the orientation of the scene, i.e. the position of the distractor.

We then created Scenes 6–10 by changing the spatial relation and orientation in each of the first five scenes. For example, Scene 8 was generated from Scene 3 by placing the target in front instead of on top of the landmark and flipping the scene, so that the distractor is on the right instead of the left. Scenes 1–10 constitute Trial Set 1.

The second trial set containing Scenes 11–20 was generated from the first one by changing the colour template and again flipping each scene along the vertical middle axis.

Because all 20 stimuli were generated from the same five schemata, they naturally fall into five different *groupings*. Due to the systematic generation process, we ensured that the target–landmark relation, the orientation and the colour template of the scenes within each *grouping* never fully coincide: if two scenes share one characteristic (e.g. the colour template), then they differ on the other two (in that case, orientation and target–landmark relation).

3 The GRE3D3 Corpus[1]

3.1 Data Normalisation and Filtering

One of the 74 participants asked for their data to be discarded. We also disregarded the data of one other participant who reported to be colour-blind. One participant consistently produced very long and syntactically complex referring expressions including reference to parts of objects and the onlooker, such as *the red cube which rests on the ground and is between you and the yellow cube of equal size*. While these descriptions are very interesting, they are clearly outliers in our data set.

Eight participants consistently only used type to describe the target object, for example simply typing *cube* for the target in Scene 5. These descriptions were excluded from the corpus under the assumption that the participants had not understood the instructions correctly or were not willing to spend the time required to type fully distinguishing referring expressions for each trial. After removal of this data, we have 630 descriptions: 30 for each of the ten scenes from Trial Set 1, and 33 for each scene in Trial Set 2.

Before conducting any quantitative data analysis, we carried out some syntactic and lexical normalisation. In particular, we corrected spelling mistakes; normalised names for *colour* values and head nouns (such as *box* instead of *cube*); and replaced complex syntactic structures such as relative clauses with semantically equivalent simpler ones such as adjectives. These normalisation steps should be of no consequence to our analysis, as we are solely inter-

[1]We refer to the data set resulting from the experiment described above as the GRE3D3 Corpus; the name stands for '**G**eneration of **R**eferring **E**xpressions in **3D** scenes with **3** Objects'. The corpus is available online at http://www.ics.mq.edu.au/∼jviethen/spatial/index.html.

ested in exploring the semantic content of referring expressions, not their lexical and syntactic surface structure.

3.2 Object Properties Used in the Corpus

The non-relational properties that occur in the corpus to describe the target object are limited to the expected type, colour and size. In five cases size was used as a relation between two objects in the form of *the same size as* or *larger than*. The spatial relations used are also mainly the expected on-top-of and in-front-of relations; however, left-of and right-of were used four and six times respectively, to relate the distractor object to one of the other two.

In addition to these direct and spatial relational properties, some participants used locative expressions such as *to the left*, *to the right*, *in the front* and in the top to describe the objects. Strictly speaking, these express relations to regions of the scene; however, we treat them as a distinct class of properties, which allows us to study the use of spatial relations between two objects separately from the use of relations to the scene.[2]

4 Data Analysis

In this section, we first provide a general analysis of the GRE3D3 Corpus data in Section 4.1. Then, in Section 4.2, we look at how characteristics of the overall scene impact on the forms of reference that people choose to use; and in Section 4.3, we look at how similarities between the objects in the scene impact on the forms of reference chosen.

4.1 General Analysis

Despite the fact that spatial information was not necessary in any of the trials for full object identification, 224 (35.6%) of the 630 descriptions contained a spatial relation to the landmark. 10 descriptions also contain a spatial relation to the distractor object. An example of a relational description that was given for the target in Scene 1 is *the green ball on top of the blue cube*.

Colour is used in 497 (78.9%) of all scenes. In 141 of these it is used twice, once for the target and once for the landmark; and in the ten descriptions that make reference to the distractor, it is mentioned for all three objects. Size is used considerably less frequently, in only 288 (45.7%) of all descriptions. Only 43 descriptions use size for both target and landmark; and five mention size for all three objects.

62 relations to regions of the scene were found in the corpus, as in *the blue cube in the front* for Scene 10. However, 74.2% of these locative expressions described the landmark and not the target referent itself. Four descriptions contained two relations to regions of the scene. Note that we concentrate our analysis here on spatial relations between actual objects and exclude relations to regions of the scene.

As noted in (Viethen and Dale, 2008), some participants used relations for all 10 scenes presented to them, and some never used relations. Not counting relations to regions of the scene, 9 participants opted to always use relations, and 24 adopted a relation-free strategy.

Figure 3 shows a falling trend in the use of relations from the first scenes the participants saw to the later ones. On the basis of subjects' comments provided on completion of the experiment, we believe that this is a kind of 'laziness effect', whereby subjects noticed after a few trials that relations were unnecessary and stopped using them. This suggests that the use of relations would potentially be even higher than one third in a setting where no such laziness effect could occur, such as the first mention of an object in a real-world situation.

4.2 The Impact of Scene Characteristics on Referring Expressions

4.2.1 Target–Landmark Relation

Recall that the target can either be on top of the landmark or in front of it, with half of our scenes (the odd-numbered ones) being of the first type, and half (the even-numbered ones) being of the second type.

In (Viethen and Dale, 2008) we observed that the number of relational descriptions in the GRE3D3 Corpus is much higher for scenes with an on-top-of relation than for those with in-front-of relations between target and landmark: 63.6% of relational descriptions are used in on-top-of scenes, while only 36.4% are used in scenes where the target is in front of the landmark.

However, in that earlier analysis, we counted the

[2]Note that in (Viethen and Dale, 2008) relations to regions of the scene were treated the same as relations to other objects.

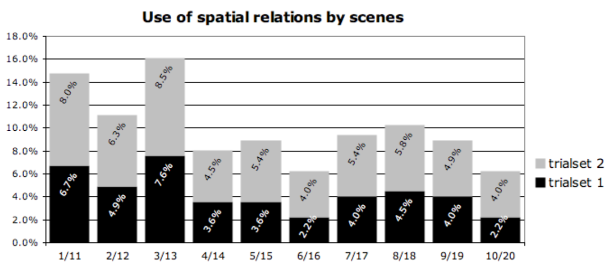


Figure 3: The use of relations for each scene. Scenes that only differ in colour template and scene orientation are stacked.

use of all locative expressions, including those involved in reference to parts of the scene as well as to landmar objects. Excluding such confounding data from the analysis confirms that if people chose to use a spatial relation, they did in fact use on-top-of relations in 'on-top-of' scenes, and in-front-of relations in 'in-front-of' scenes. This investigation also verifies the preference for on-top-of over in-front-of. Overall, 58.0% of the 224 relational descriptions contain an on-top-of relation between target and landmark, while only 42.0% use in-front-of relations, which is a statistically significant difference (χ^2=8.98, df=2, $p < .05$). Expressed differently, this means that of the 315 trials where the scene contained an on-top-of relation between target and landmark, on-top-of was used in 41.2% of the instances, and in-front-of was only used in 29.8% of the 315 'in-front-of' trials (χ^2=5.79, df=2, $p < .05$).

This uneven distribution of spatial information between the two types of target–landmark relations could be due to the fact that, in 'in-front-of' scenes, the landmark is partially occluded by the target object. Another factor that might be at play is people's general preference for relations on the frontal axis of a landmark over those on the lateral axis (see Tenbrink (2005), p. 18).

4.2.2 Colour Templates and Scene Orientation

As expected, the colour template used in a scene did not have a significant effect on whether relations between objects were used, whether colour was used, or whether a relation to a region of the scene was used; in each case, the difference between the two colour templates is one percentage point or less.

Similarly, whether the distractor was displayed to the left or the right of the landmark had no significant effect on the content of the referring expressions produced.

4.3 The Impact of Object Similarity on Referring Expressions

As mentioned in Section 2.2, the psychological literature on visual salience suggests that the visual salience of an object is mainly determined by how much its appearance differs from that of its surrounding objects. We attempted to keep other factors, such as the distance between objects and occlusion of objects, as constant as possible between scenes, so that an object becomes more salient as the number of properties it shares with the other two objects decreases, and as the similarity of those other two objects to each other increases. We therefore report below results from the analysis of the extent to which the similarity between target, landmark and distractor influences the use of relations.

For the analysis of reference behaviour in the simple scenes we used, it is sufficient to adopt this rough notion of visual salience. However, it is not clear how it would carry across to more complex scenes with more properties and more objects. For example, it is not obvious whether an object that shares its type with all other objects, but is unique in all other properties, is more salient than an object that is unique in type but shares all other properties with one object each.

4.3.1 Target Salience

In order to answer Question 1 from Section 2.2.2 (whether the use of relations is impacted by the length of the *minimal non-relational description* for the target), we test for the influence of the number of properties the target shares with any other object in the scene as a rough measure of its visual salience.

In Schemata A and B, the target never shares its type with one of the other objects, which means it can be described uniquely by using only type. In Schemata C and E, it shares two properties with another object, so the length of its minimal description is two. While not being identical with either object in Schema D, here the target shares all three property values with one of the other objects and therefore can only be distinguished by a referring expression of at least length three.

The use of relations to the landmark drops slightly for targets that are very similar, i.e. less visually salient, from 36.9% to 30.2%. Although this drop is not statistically significant, the trend is surprising. If the trend is confirmed in future experiments, it might be explained by a hypothesis similar to that proposed by Edmonds (1994): the information giver's confidence has to be high enough to satisfy them that they have conveyed sufficient information. In addition to the requirement that the description has to be fully distinguishing, having to meet such a confidence threshold might compel people to add visually salient information, such as spatial relations to prominent landmarks, to a short referring expression, while the threshold might already be satisfied by longer non-relational descriptions.

4.3.2 Target–Landmark Similarity

Since we excluded scenes where the landmark is a ball, the test for influence of target type coincides with the one that can help us answer Question 2 from Section 2.2.2: Is the use of relations impacted by the similarity between target and landmark? The way in which we have defined visual salience means that the similarity between two objects impacts the visual salience of both in the same way. We expected that a visually salient target would receive fewer relational descriptions, while a visually salient landmark would result in relations being used more often, so this test can also give us an idea of which object's visual salience yields more influence on the use of relations. In scenes based on Schemata D and E, target and landmark are similar, while they are completely distinct in those based on Schemata A, B and C.

This factor had a clear effect on the use of all properties in the corpus: For scenes with dissimilar target and landmark, the use of spatial relations was at 1.4 times significantly higher (40.2% of all descriptions for these scenes) than for the scenes where they were similar (28.6%) (χ^2=8.94, df=1, $p < .01$); the use of the non-relational properties (colour and size), on the other hand, was much lower. This outcome, supported by the non-conclusive results of the previous section, suggests that, at least in our domain, the visual salience of the landmark has more impact on the choice of whether a relation between it and the target gets included in the referring expression than the visual salience of the target itself.

4.3.3 Landmark Salience

If the visual difference between the objects in a scene is the main factor determining whether and which spatial information gets used, then it stands to reason that the visual salience of a potential landmark (i.e., how different it is from the surrounding objects) has a particularly high influence on whether the specific relation between it and the target object gets used. In (Viethen and Dale, 2008) we made a weak claim to this effect based on the fact that particularly many relational descriptions were recorded for scenes based on Schema C, where the landmark differs from the other two objects in type and colour. More detailed analysis of the data reveals that, indeed, the more different the landmark is from the other two objects, the more likely it is to be included in the referring expression via a spatial relation.

In 46.8% of all trials where the scene contained a landmark very dissimilar to both other objects (Schema C), a spatial relation to the landmark was used. In scenes where the landmark was more similar to the distractor than the target (Schemata A and B), this number fell to 36.9%. Landmarks sharing more properties with target than distractor (Schemata D and E) were included in referring expressions in only 28.6% of trials where they occurred. The difference between these three categories is statistically significant at χ^2=12.55 (df=2, $p < .01$).

Considering that the target object is already in focus when the landmark's salience is evaluated, the visual difference between the landmark and the target might be of more importance. This is in line with the findings in the previous section.

Below we look at three individual factors that influence the landmark's visual salience; the difference of its size and colour from the other two objects and its overall difference from the distractor in all properties.

Landmark Size Interestingly, the percentage of relational descriptions is significantly lower for scenes where the landmark has a unique size (33.3%) than for those where it shares its size with the distractor object (42.1%). For scenes where the landmark has the same size as the target, the number is significantly lower again (27.0%), as expected (χ^2=9.24, *df*=2, $p < .01$). However, these numbers bear the influence of both the 'laziness effect' reducing relation use for scenes displayed late and the fact that in half of the trials where landmark and distractor had the same size, the landmark was unique in its two other properties, colour and type.

Landmark Colour The trend for the influence of the landmark's difference in colour to the other objects is similar, although not as pronounced as for size and not statistically significant: landmarks with unique colour are used in spatial relations slightly less often (36.8%) than those sharing their colour with the distractor (37.3%), while spatial relations with landmarks sharing the colour of the target are only used in 30.2% of the trials where this occurred.

Interestingly, the use of the landmark's colour shows the same trend as the influence of colour difference on the use of a landmark: it is most likely to be actually mentioned in relational descriptions for scenes where it is the same as the colour of the target object (94.7%). In relational descriptions for scenes where the landmark had a unique colour, this colour was mentioned in only 73.4% of the cases. Landmark colour was only verbalised in 48.9% of the relational descriptions where it was the same as the colour of the distractor. These differences are statistically significant at χ^2=11.45 (*df*=2, $p < .01$). This data can help us determine which properties to mention for a landmark, once the decision to refer to landmark has been made.

Landmark–Distractor Similarity We attempt to answer Question 3 from Section 2.2.2 by testing for an effect of the visual salience of the distractor on referring expressions. Visual salience here is again effectively the inverse of the number of properties shared with other objects in the scene.

The use of spatial relations is at its lowest for scenes where the distractor is completely distinct from the landmark object (Schema E). 27.0% of all descriptions for these scenes contained relations. While this difference is not statistically significant, we had expected an opposite trend due to the distinctness of the distractor from the landmark in these scenes making the target–landmark cluster appear as a more salient unit, compelling people to use the spatial relation between them. This was not the case; but again this might be due to the influence of the laziness effect having set in.

There was almost no difference between the usage of relations for scenes where landmark and distractor are identical (based on Schema A) and those where they share one property value (based on Schemas B, C, D). When they were identical relations were used in 37.3%, otherwise in 37.8% of all scenes in either condition.

5 Conclusions

The data analysis presented above clearly suggests that all scenes are not the same when it comes to the use of spatial relations in referring expressions. It brings up the interesting problem of how to model the apparent preference of our participants to use relations in some scenes more than in others. Following our discussion above, the factors that lead to this preference seem to be determined by the visual salience of the different objects involved. In particular, the visual salience of a potential landmark seems to be of high importance. The specific factors we found to have an impact include:

- the type of spatial relation that holds between the target and a potential landmark;
- the visual salience of the target as measured by the number of properties it has that are not shared with other objects around it;
- the visual salience of the landmark as measured by its inverse similarity to the target referent.

Factors like these can be incorporated into a referring expression generation algorithm by taking them into account in the step that calculates which property of the target object should next be considered for inclusion in the referring expression. A preference score for each property needs to be determined 'at run time', which would also allow the consideration of the *discourse* salience of a property, which results from previous mentions and the purpose of the discourse. The preference scores of the properties in a referring expression under construction would then combine into an adequacy score for the overall description.

In future work we intend to embody these observations in an implemented algorithm for the generation of referring expressions.

References

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22:1482–1493.

Donna K. Byron and Eric Fosler-Lussier. 2006. The OSU Quake 2004 corpus of two-party situated problem-solving dialogs. In *Proceedings of the 15th Language Resources and Evaluation Conference*.

David Caduff and Sabine Timpf. 2007. On the assessment of landmark salience for human navigation on the assessment of landmark salience for human navigation. *Cognitive Processes*.

Robert Dale and Nicolas Haddock. 1991. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265. Robert?

Robert Dale, Sabine Geldof, and Jean-Philip Prost. 2005. Using natural language generation in automatic route description. In *Journal of Research and Practice in Information Technology*, volume 37, pages 89–105.

Philip G. Edmonds. 1994. Collaboration on reference to objects that are not mutually known. In *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Claire Gardent. 2002. Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, USA.

Peter Gorniak and Deb Roy. 2004. Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research*, 21:429–470.

Pamela W. Jordan and Marilyn A. Walker. 2005. Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research*, 24:157–194.

John Kelleher and Geert-Jan M. Kruijff. 2005. A context-dependent model of proximity in physically situated environments. In *Proceedings of the 2nd ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, Colchester, U.K.

John Kelleher and Geert-Jan M. Kruijff. 2006. Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st COLING and the 44th ACL Conference*, Sydney, Australia.

Emiel Krahmer and Mariët Theune. 2002. Efficient context-sensitive generation of referring expressions. In Kees van Deemter and Rodger Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI Publications, Stanford, CA.

Harold Pashler. 1998. *Attention*. Psychology Press, Hove, UK.

Thora Tenbrink. 2005. Semantics and application of spatial dimensional terms in English and German. Technical Report Series of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition, No. 004-03/2005, Universities of Bremen and Freiburg, Germany.

Henry S. Thompson, Anne Anderson, Ellen Gurman Bard, Gwyneth Doherty-Sneddon, Alison Newlands, and Cathy Sotillo. 1993. The HCRC map task corpus: natural dialogue for speech recognition. In *Proceedings of the 1993 Workshop on Human Language Technology*, pages 25–30, Princeton, New Jersey.

Sebastian Varges. 2005. Spatial descriptions as referring expressions in the maptask domain. In *Proceedings of the 10th European Workshop On Natural Language Generation*, Aberdeen, UK.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of the 5th International Conference on Natural Language Generation*, Salt Fork OH, USA.

Steven Yantis and Howard E. Egeth. 1999. On the distinction between visual salience and stimulus-driven attentional capture. *Journal of Experimental Psychology: Human Perception and Performance*, 25(3):661–676.

Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 169-177

Morphosyntactic Target Language Matching in Statistical Machine Translation

Simon Zwarts
Centre for Language Technology
Macquarie University
Sydney, Australia
szwarts@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
Sydney, Australia
madras@ics.mq.edu.au

Abstract

While the intuition that morphological preprocessing of languages in various applications can be beneficial appears to be often true, especially in the case of morphologically richer languages, it is not always the case. Previous work on translation between Nordic languages, including the morphologically rich Finnish, found that morphological analysis and preprocessing actually led to a decrease in translation quality below that of the unprocessed baseline.

In this paper we investigate the proposition that the effect on translation quality depends on the *kind* of morphological preprocessing; and in particular that a specific kind of morphological preprocessing before translation could improve translation quality, a preprocessing that first transforms the source language to look more like the target, adapted from work on preprocessing via syntactically motivated reordering. We show that this is indeed the case in translating from Finnish, and that the results hold for different target languages and different morphological analysers.

1 Introduction

In many NLP applications, morphological preprocessing such as stemming is intuitively felt to be important, especially when dealing with morphologically rich languages. In fairly early work on morphological disambiguation of the agglutinative language Turkish, Hakkani-Tur et al. (2000) cite the note of Hankamer (1989) that a single Turkish root can potentially have over a million inflected variants; such a proliferation of forms could exacerbate any data sparsity problems. Zwarts and Dras (2007a) showed, for languages that differ in morphological richness but are otherwise structurally similar, that, as expected, morphological preprocessing in an Statistical Machine Translation (SMT) environment confers greater benefits on the morphologically richer language than on the morphologically poorer one.

However, it is not always the case that morphological preprocessing of a morphologically rich language does provide a benefit. Again in an SMT context, Virpioja et al. (2007) preprocessed Finnish (a Uralic language, typologically between inflected and agglutinative, and consequently morphologically rich) for translation into Danish and Swedish (both Indo-European languages, quite different in many morphosyntactic respects from Finnish, including being morphologically less rich). In doing this, they found that translation quality was generally worse than baseline, and never better.

What this paper looks at is whether it is the way in which the morphology is preprocessed that is important. We draw on an idea from other work in machine translation first presented by Collins et al. (2005), where the source language is reordered so that its syntax is more like that of the target language, leading to an improvement in translation quality; here, we see whether that idea can apply at the level of morphology. In particular, where there are phenomena that are handled via morphology in one language and by syntax in another, we investigate whether an adaptation of this reordering idea can improve translation quality.

In Section 2 we briefly review relevant literature. In Section 3 we describe the important characteristics of Finnish, followed by various models for morphological preprocessing, including our method for transforming the morphology of the Finnish source to more closely match the target language. In Section 4 we discuss the results, and in Section 5 we conclude.

2 Literature Review

2.1 Morphological Analysis

So many systems make some use of morphological preprocessing, particularly stemming, that we only point to a few specific instances here. In the context of parsing, morphological preprocessing has been shown to be necessary for the agglutinative languages Korean (Han and Sarkar, 2002) and Turkish (Eryiğit and Oflazer, 2006). In SMT, use of morphological preprocessing has been fairly ad hoc. One quite systematic comparison of morphological preprocessing parameters was carried out by Sadat and Habash (2006) for the language pair Arabic-English; their approach was just to search the whole space of parameter combinations, rather than looking at any characteristics of the pairing of the specific languages. One earlier work looking at Czech-English, Al-Onaizan et al. (1999), did carry out morphological analysis that looked at characteristics of the pair of languages, transforming some Czech morphemes into pseudo-prepositions.

The specific work we cited in the Introduction as having found that morphological preprocessing did not help, that of Virpioja et al. (2007), used Morfessor, a morphological analyser for Nordic Languages, as a preprocessor. They built translation systems between the Nordic languages Danish, Finnish and Swedish. They found that using Morfessor's morphemes as translation units instead of words degraded SMT quality[1] for all six language pairs and language directions possible (four statistically significantly). They tried unsupervised learning techniques to decide when to use morphological information and when to use normal phrases; this helped the system, but did not manage to beat a normal Phrase-Based Statistical Machine Translation (PSMT) system using words as the most fine-grained translation units.

[1]Tables 6 and 7 in their paper

2.2 Source-Side Reordering as Preprocessing

There are a number of different approaches to word reordering. It can be done based on rules over word alignment learnt statistically, for example Costa-Jussà and Fonollosa (2006). In this work an improvement in overall translation quality in a Spanish-English MT system was achieved by using statistical word classes and a word-based distortion model to reorder words in the source language. Reordering here is purely a statistical process and no syntactic knowledge of the language is used. Xia and McCord (2004) on the other hand use syntactic knowledge; they use pattern learning in their reordering system. In their work they parse and align sentences in the training phase and derive reordering patterns. From the English-French Canadian Hansard they extract 56,000 different transformations for translation. In the decoding phase they use these transformations on the source language. The main focus then is monotonic decoding. Both of these two cited works assume that explicitly matching the word order of the target language is the key.

The work that we draw on in this paper is that of Collins et al. (2005), which uses syntactically motivated rules based on clause restructuring. They define six hand-written rules for reordering source sentences in German for translation to English, which operate on the output of an automatic parser. The rules cover German phenomena such as the location of verbs, separable verb prefixes, negations and subjects, several of which represent long-distance relationships in German (e.g. where the inflected verb is in second position in the clause and its uninflected dependendent verbs are at the end of the clause). The approach has also been applied successfully to Dutch-English (Zwarts and Dras, 2007b) and Chinese-English (Wang et al., 2007), among others.

Zwarts and Dras (2007b) found that there are at least two sources of the translation improvement: one is the explicit matching of target language syntax, while the other is the moving of heads and dependants closer together to take advantage of the phrasal window of PSMT.

3 Morphological Target Language Matching

3.1 Languages

Taking the work of Virpioja et al. (2007) as a broad starting point, we use Finnish as our source language. As noted above, Finnish is part of the Finno-Ugric branch of the Uralic language family rather than part of the majority Indo-European family, and compared to languages like English is very rich in morphology. Finnish has fifteen noun cases: four grammatical cases, six locative cases, two essive cases and three marginal cases. It has three verb moods and on the imperative mood Finnish marks: 1st, 2nd or 3rd person, singular or plural, definite or indefinite and positive or negative. Verbs can have a morphological perfect, present or future tense. Finnish needs morphological agreement between words, for example noun and adjective agreement. All of this morphology is purely postfixing. The case system, which will be the focus in this paper, is described in more detail in Table 1.

For our main target language we use English, which is morphologically not at all a rich language. As a supplementary target language, to check results for Finnish-English, we use Dutch, which is morphologically quite similar to English, in terms of quantity and type of inflection.

3.2 Models

We hypothesise that an important reason for a gain in translation quality when using morphological preprocessing is the matching of the morphology and syntax of the target language. As in the work described in Section 2.2, where the source language was made to look more like the target language by applying grammatical rules, now we want to do this on a morphological level. As comparisons, we design several models which all differ in the fact that they have a different way of preprocessing the text.

To obtain morphological analyses we use Connexor[2] (Tapanainen and Järvinen, 1997) and Morfessor[3] (Creutz et al., 2005) for Finnish. Connexor provides a per-word morphological analysis and it provides the stem of the tokens. Connexor indicates the Part-Of-Speech (POS) and depending on that other information. For example for a noun it gives the case, and whether it is singular or plural. It does not indicate morpheme boundaries; however, since it provides stems, boundaries are recoverable if the token in question has multiple stems. Connexor also provides other parse information, but since we are interested in morphology in this paper, we only use the morphological information seen in the previous example. Morfessor, on the other hand, only indicates morpheme boundaries and indicates per morpheme whether this morpheme is a suffix or a stem. Figure 1 shows an example of the first line in the Europarl Corpus.

We build models with different amounts of preprocessing, and different types of preprocessing, and investigate their effects: four models based on Connexor output, one straight PSMT baseline model, and two models based on Morfessor output. We first define the main model of interest, where the morphology of the source language is mapped to the morphology and syntax of the target language. Then as comparisons we look at baselines with no morphology, preprocessing with full morphology, and preprocessing with word stems only, both with compounds and without.

Model C1 - Noun Case matching This is the model with specific morphological matching to the target language. Finnish has many different noun cases, while the case system has almost completely disappeared in English and Dutch, as only the pronouns still exhibit some vestiges of a case system. Finnish however has many cases, even compared to other European languages which still have case systems (for example German, Greek etc.). A lot of cases in Finnish fall in the group of locative case, which is to indicate how the noun is located. In English and Dutch this is usually expressed with prepositions.

The preprocessing steps are then as follows:

[2]`http://www.connexor.com`

[3]`http://www.cis.hut.fi/projects/morpho/`

Finnish cases

Case	Suffix	English prep.	Sample	Translation
		Grammatical		
nominatiivi (nominative)		-	talo	house
genetiivi (genitive)	-n	of	talon	of (a) house
akkusatiivi (accusative)	- or -n	-	talo or talon	house
partitiivi (partitive)	-(t)a	-	taloa	house (as an object)
		Locative (internal)		
inessiivi (inessive)	-ssa	in	talossa	in (a) house
elatiivi (elative)	-sta	from (inside)	talosta	from (a) house
illatiivi (illative)	-an, -en, etc.	into	taloon	into (a) house
		Locative (external)		
adessiivi (adessive)	-lla	at, on	talolla	at (a) house
ablatiivi (ablative)	-lta	from	talolta	from (a) house
allatiivi (allative)	-lle	to	talolle	to (a) house
		Essive		
essiivi (essive)	-na	as	talona	as a house
(eksessiivi; dialectal) (exessive)	-nta	from being	talonta	from being a house
translatiivi (translative)	-ksi	to (role of)	taloksi	to a house
		Marginal		
instruktiivi (instructive)	-n	with (the aid of)	taloin	with the houses
abessiivi (abessive)	-tta	without	talotta	without (a) house
komitatiivi (comitative)	-ne-	together (with)	taloineni	with my house(s)

Table 1: Finnish Case system, taken from Wikipedia. (`http://en.wikipedia.org/wiki/Finnish\_grammar`)

Finnish:	istuntokauden uudelleenavaaminen
Connexor:	Lemma=‘istunto kausi’ Morpho=‘N SG GEN’
	Lemma=‘uudelleen avata’ Morpho=‘V ACT INF4 NOM’
Morfessor:	istu/STM n/SUF tokauden/STM
	uude/STM lle/SUF en/SUF avaam/STM in/SUF en/SUF
English:	resumption of the session

Figure 1: Morphological Output from Connexor and Morfessor

1. For every token in the sentence we retrieve the POS and case information.
2. For every token marked as `N` (noun) we replace this token by its stem.
3. For every token marked as `N` we replace all tokens directly preceding the noun by its stem if this token shares the same case marker as the noun.
4. We insert before each initial token in Step 3 a token marking the case.

To explain: To make the Finnish more like the target language, we need to remove case morphology from the noun itself as the target language does not have cases morphologically marked. Therefore in Step 2 we replace the token by its stem. Because Finnish is a language with agreement, if a token has a noun case, other tokens in agreement with this noun (for example adjectives) need to undergo the same case-morphology removal step; this is Step 3. Finally in the last step we need to insert a token, providing the information of case as a separate token. Usually, this will result in being translated as a preposition. This information is represented in a token before the actual noun, matching the English word ordering where the preposition is positioned before the noun. We chose to introduce this token as `-T-`<case> so that for a genitive case, for example, we introduce the token `-T-GEN`. Investigation shows that in this way there is no clash between these tokens and Finnish vocabulary. An example is provided in Table 2, showing the original Finnish, the preprocessed variant and an English reference.

We note that in general we would not expect this to move related morphemes into or out of the phrasal window used in PSMT, thus we would not expect to gain anything as a consequence of this source of translation improvement described in Zwarts and Dras (2007b). However, the other source of improvement, the explicit matching of the target language, is still possible.

Model C2 - Full morphological preprocessing Part of our hypothesis is that it is not full morphological preprocessing which will bring the most improvement of translation quality, but rather the morphological matching of the target language; so we construct a full preprocessing model for comparison. For our full morphological preprocessing model, we replace every token by its stem and insert after every token a token indicating morphology. For example the Finnish word *istuntokauden* (‘resumption’, from Figure 1) with Lemma=‘istunto kausi’ and Morpho=‘N SG GEN’ will result in the three tokens: *istunto kausi N-SG-GEN*. This is the same basic approach as that of Virpioja et al. (2007).

Model C3 - Stem Only - Compounds In this model we investigate the effects of having no morphology in Finnish at all. Because Finnish is morphologically much richer than English, there are many more individual tokens in Finnish with the same base stem, while in English translation often only one token is there to express this concept. To make the Finnish language less sparse, and make it possible to do a more reliable alignment estimation from which we calculate probabilities for Finnish to English, we only use stems. For example, the Finnish word *vaunu* (‘car’) can take many different surface realisations depending on morphology (*vaunut, vaunuilla, vaunuja, vaunujen, vaunujensa* etc.). Without morphological preprocessing these are entirely different tokens. We map all these tokens onto the same base stem *vaunu*, so the probability estimation for *car* and *vaunu* should be more accurate. In this model we leave compounds (for example compound nouns) as one token. The word *istuntokauden* results in the token *istuntokausi*.

Model C4 - Stem Only - Compounds separated This model aims to investigate the effect of compound splitting with stemming. This model is similar to the previous model, Model C3, except now we do split compounds. The word *istuntokauden* results in the two tokens *istunto* and *kausi*.

Koehn (2003) shows that compound splitting in Machine Translation (MT) is superior to leaving compounds intact. However, for completeness we include both models.

Model B - Baseline, no morphological preprocessing Here we just use the straight PSMT output, taking the original source as the input.

Original Finnish	istuntokauden uudelleenavaaminen
Case Isolated Finnish	-T-GEN istuntokausi uudelleenavaaminen
English Reference	resumption of the session
Original Finnish	äänestimme -T-ELA asia -T-GEN keskustelu jälkeen
Case Isolated Finnish	äänestimme asiasta keskustelun jälkeen
English Reference	we then put it to a vote
Original Finnish	-T-INE ensimmäinen käsittely tekemämme -T-NOM tarkistusehdotus on otettu -T-POSS:SG1 mieli kiitettävästi huomioon
Case Isolated Finnish	ensimmäisessä käsittelyss tekemämme tarkistusehdotukset on otettu mielestäni kiitettävästi huomioon
English Reference	our amendments from the first reading have i believe been taken into account very satisfactorily

Table 2: Noun Case Matching

Model M1 - Noun Case matching We also experimented with a different type of morphological information, to see if our findings still held up. Morfessor, as discussed, provides different information from Connexor. The baseline model (Model B) is the same for the Connexor models and the Morfessor models since in both cases it does not have any preprocessing. Other models are designed to target different morphological analysis.

This model is targeted to be the Morfessor equivalent of Model C1 for Connexor. The morphological preprocessing is targetted to match the morphology of the target language. As in the Connexor model we target the elaborate locative case system in the Finnish language. We identify which morphological suffixes are responsible for indicating a noun case. These morphemes[4] are the only morphemes treated. By default we leave a token unchanged by the preprocessing algorithm. However if one of these morphemes appears in the token, we delete this morpheme from the token (with all its doubles, because we assume these are there for agreement reasons) and put that morpheme in front of the token we are preprocessing. On the suffix we leave the suffix identifier attached so these tokens do not clash with other words in the Finnish vocabulary.

[4]The complete list is: n/SUF, ssa/SUF, sta/SUF, en/SUF, an/SUF, lla/SUF, lta/SUF, lle/SUF, na/SUF, nta/SUF, ksi/SUF, tta/SUF, ne/SUF. See Table 1 for how these suffixes are acquired.

For example:

istuntokauden uudelleenavaaminen

turns into:

n/SUF *istutokauden lle*/SUF *en*/SUF *uudeavaamin*

Model M3 - Stemmed This model is the Morfessor equivalent of Connexor model C3. We directly input the output from Morfessor into the PSMT system. Morfessor identifies the different morphemes in the different tokens. For each morpheme Morfessor identifies whether it is a suffix or a stem morpheme. We leave this identifier attached to the morpheme.

For example:

istuntokauden uudelleenavaaminen

turns into:

istu/STM n/SUF *tokauden*/STM *uude*/STM *lle*/SUF *en*/SUF *avaam*/STM *in*/SUF *en*/SUF

3.3 Data

We use as our data the Europarl corpus, language pairs Finnish-English and Finnish-Dutch. In both cases the baseline is a normal sentence-aligned corpus, which is trained with GIZA++. After this training, the phrases are derived via the standard method described in the Pharaoh manual (Koehn, 2004). The language model is trained on Europarl text only and is an n-gram language model. We use Pharaoh to translate a test set of 10k sentences while we have 774k sentences in the training corpus.

The same approach is used for the various other models, with the text passed to GIZA++ and Pharaoh being the preprocessed version of the source. As Virpioja et al. (2007), we do not perform Minimum Error Rate Training (MERT) (Och, 2003) or optimise variables.

4 Results and Discussion

The results, expressed in BLEU points, are shown in Table 3 for Finnish to English. For the translations to Dutch with the Connexor models, results are shown in Table 4, which follow the trends of the English models.

First of all, we note that the the poor performance of the full morphological analysis using Morfessor, Model M3, relative to the baseline, confirms the findings of Virpioja et al. (2007).

However, as can be seen from the tables, it is possible to outperform the baseline, but only where the specially targetted morphological analysis has been performed. This is true both where the morphological analysis has been done using Connexor (Model C1) and where it has been done using Morfessor (Model M1). All of the other preprocessings perform worse. This result also carries over to Dutch.

We had a closer look at Model B and Model C1, the baseline and the model with special targetted morphological analysis. Table 5 shows how the BLEU score is calculated for the baseline and the model where morphology has improved translation quality. We can see that in fact the token-for-token translation quality is not that different. The brevity penalty is what is responsible for most of the gain, which suggests that the original baseline is undergenerating words.

It is possible that the large brevity penalty is

Finnish	**to English**
B. Original (baseline)	0.1273
Connexor	
C1. Noun Case Isolated	0.1443
C2. Stemmed, separate morph.	0.1082
C3. Stemmed	0.1060
C4. Stemmed and de-compounded	0.1055
Morfessor	
M1. Noun Case Isolated	0.1312
M3. Stemmed	0.1073

Table 3: BLEU scores for different morphological analysis

Finnish	**to Dutch**
B. Original (baseline)	0.132
Connexor	
C1. Noun Case Isolated	0.138
C2. Stemmed, separate morph.	0.089
C3. Stemmed	0.101
C4. Stemmed and de-compounded	0.102

Table 4: BLEU scores for different morphological analysis

	Model B	**Model C1**
1-gram	0.5167	0.4945
2-gram	0.2276	0.2135
3-gram	0.1114	0.1029
4-gram	0.0564	0.0500
Brevity Penalty	0.7717	0.9453
BLEU	0.1273	0.1443

Table 5: Decomposition of BLEU scores for Model B and Model C1

a consequence of not carrying out MERT.[5] Och (2003) notes that using standard optimisation criteria (rather than optimising for BLEU score) can "prefer shorter translations which are heavily penalized by the BLEU and NIST brevity penalty". However, all of the models are comparable, being implemented without MERT, so they are all affected in the same way.

In attempting to understand it, we note that what the morphological preprocessing has achieved is to generate more correct words. We can conclude that in the normal (baseline) translation the problem is not so much to find the right translation for the words: even when morphology is attached the decoder still manages to find right translations, but fails to decode that part of information with morphology holds by itself.

Even PSMT is based on underlying token-to-token translation tools. Although the phrases can handle translation of multiple tokens to multiple tokens, the most common ways to derive these phrases are still based on single token to token probability estimation. GIZA++, which is used for phrase extraction, needs fertility to translate one token to multiple tokens in the target language. In our Finnish corpus, every sentence has on average 14.0 tokens per sentences against 20.3 for English. With the model C1's preprocessed text (see Table 2) we have on average 18.4 tokens per sentence, which is much closer to the target language. So an explanation is that this closer matching of language sizes has led to this difference.

5 Conclusion

We have investigated whether the idea of reordering as preprocessing carries over to the interface between morphology and syntax. We have compared this to a series of models covering a range of morphological preprocessing, and shown that among these the reordering model, restructuring the morphology of the source language to look more like the target language, works the best, and is in fact the only model to outperform the baseline. In particular this model outperforms the model based on previous work where it is full morphological information that is added in the preprocessing step. These results hold across the target languages English and Dutch, and for two different morphological analysers.

In terms of future work, Section 4 raised the issue of MERT; the next step will involve looking at the effect, if any, of this. With respect to directions subsequent to this, the work in this paper targetted only one particular phenomenon on the interface of morphology and syntax, the correspondence of locative cases in Finnish to prepositions in English and Dutch. There are many potential other phenomena that could also be captured, and possibly combined with a broader class of reorderings, an area for future investigation. In addition, our work did not use any of the unsupervised learning methods that Virpioja et al. (2007) used to decide when to use morphological analysis, which for them improved results (although only up to the baseline); there would be scope to apply a similar idea here as well, perhaps adapting the work of Zwarts and Dras (2008) in identifying situations where syntactic reordering has led to a worse translation in the case of an individual sentence.

[5]Thanks to an anonymous reviewer for this suggestion.

6 Acknowledgements

We would like to thank Tuomo Kakkonen, for helping us with Finnish advice and for supplying us with an automated morphological markup of our training text.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation, final report, JHU Workshop, Dec.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 531–540, Ann Arbor, Michigan, June.

Marta R. Costa-Jussà and José A. R. Fonollosa. 2006. Statistical Machine Reordering. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 70–76.

Mathias Creutz, Krista Lagus, Krister Lindén, and Sami Virpioja. 2005. Morfessor and Hutmegs: Unsupervised Morpheme Segmentation for Highly-Inflecting and Compounding Languages. In *Proceedings of the Second Baltic Conference on Human Language Technologies*, pages 107–112.

Gülsen Eryiğit and Kemal Oflazer. 2006. Statistical Dependency Parsing for Turkish. In *11th Conference of the European Chapter of the Association for Computational Linguistics: EACL*.

Diiek Z. Hakkani-Tur, Kemal Oflazer, and Gokhan Tur. 2000. Statistical Morphological Disambiguation for Agglutinative Languages. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, pages 285–291.

Chung-hye Han and Anoop Sarkar. 2002. Statistical Morphological Tagging and Parsing of Korean with an LTAG Grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 48–56.

Jorge Hankamer. 1989. Morphological Parsing and the Lexicon. In *Lexical Representation and Process*. The MIT Press.

Philipp Koehn. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of Association for Machine Translation in the Americas (AMTA)*, pages 115–124.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Fatiha Sadat and Nizar Habash. 2006. Combination of Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 1–8. Association for Computational Linguistics.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71.

Sami Virpioja, Jaako J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-Aware Statistical Machine Translation Based on Morphs Induced in an Unsupervised Manner. In *Proceedings of MT Summit XIII*, pages 491–498.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese Syntactic Reordering for Statistical Machine Translation. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*, pages 737–745.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling)*, pages 508–514.

Simon Zwarts and Mark Dras. 2007a. Statistical Machine Translation of Australian Aboriginal Languages: Morphological Analysis with Languages of Differing Morphological Richness. In *Proceedings of the Australasian Language Technology Workshop*.

Simon Zwarts and Mark Dras. 2007b. Syntax-Based Word Reordering in Phrase-Based Statistical Machine Translation: Why Does it Work? In *Proceedings of MT Summit XI*, pages 559–566.

Simon Zwarts and Mark Dras. 2008. Choosing the Right Translation: A Syntactically Informed Approach. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling)*.

Author Index